UNIVERSIDAD MICHOACANA
DE SAN NICOLÁS DE HIDALGO
FACULTAD DE INGENIERÍA
ELÉCTRICA

PIZARRA ELECTRÓNICA INTERACTIVA BASADA RASPBERRY PI.

TESIS

Que para obtener el título de:

INGENIERO EN ELECTRÓNICA

Presenta

URIEL TORRES DÍAZ

Asesor

M.C. ANTONIO ULISES SÁENZ TRUJILLO

Morelia Michoacán, marzo 2018.

Agradecimientos

A Dios, por brindarme la fortaleza de cumplir mis objetivos.

A mis padres María y Manuel, por apoyarme en todo momento.

A mi asesor, el M.C. Ulises Sáenz, por su tiempo y su buena disposición.

A todos los profesores y compañeros de la Facultad de los que aprendí en el transcurso.

A la Universidad Michoacana de San Nicolás de Hidalgo, sobre todo a la Facultad de Ingeniería Eléctrica.

A mis grandes amigos, con los que he compartido la experiencia de la vida.

¡A todos ellos gracias!

Dedicatorias

Este trabajo está dedicado a mis amados padres:

✞ *María Diaz Ibarra*

y

Jose Manuel Torres Reynoso

A mi madre, por motivarme día a día a ser una mejor persona y cumplir con mis metas, por sus sabios consejos y por llenarme con su amor; aunque el tiempo no nos permitió más momentos juntos, siempre llevo sus palabras en mi pensamiento y en mi corazón.

A mi padre, por apoyarme en todas mis decisiones, y sobre todo por brindarme con su ejemplo, la sabiduría, la perseverancia y la humildad que tanto admiro de él.

Resumen

Este trabajo está enfocado en el desarrollo de una herramienta flexible y de bajo costo que permita la interconexión inalámbrica entre una computadora y un proyector o pantalla, así como también la interacción y control de algunos gestos en presentaciones de diapositivas con Power Point®. El desarrollo de esta herramienta consiste en la creación de tres elementos por separado: el Marcador IR, la aplicación de escritorio y la aplicación para Raspberry Pi la cual, recibe el nombre de "Ras Pizarrón". Cada elemento de la herramienta realiza funciones distintas, las cuales son requeridas para el funcionamiento efectivo de la herramienta.

Palabras clave

Raspberry Pi, Microcontrolador, diapositiva, Marcador Ir, Ras Pizarrón.

Abstract

This work is influenced by the need to create a tool, that allows interacting with Power Point slide shows. It consists in the creation of three separate elements: the IR Marker, the Desktop Application and the Raspberry Pi application which is called of Ras Whiteboard.

Each element of the tool has different duties, which is required for the effective operation of the tool.

Keywords

Raspberry Pi, Microcontrolador, diapositiva, Marcador Ir, Ras Pizarrón.

Índice

Agradecimientos ____ ***i***

Dedicatorias ____ ***ii***

Resumen ____ ***iii***

Palabras clave ____ ***iii***

Abstract ____ ***iv***

Keywords ____ ***iv***

Índice ____ ***v***

Lista de Figuras ____ ***viii***

Lista de Ecuaciones ____ ***x***

Lista de pseudocódigos ____ ***x***

Glosario de términos ____ ***xi***

Capítulo 1 Introducción ____ ***1***

1.1 Sistemas de proyección analógicos ____ *1*

Proyector de diapositivas ____ *1*

Retroproyector ____ *2*

1.2 Sistemas de proyección digitales ____ *3*

Proyector de video ____ *3*

Pizarra interactiva ____ *4*

1.3 Sistemas de control ____ *4*

Teclado y ratón ____ *4*

Mando a distancia USB ____ *5*

Mando a distancia Infrarrojo ____ *6*

1.4 Objetivo. ____ *7*

Objetivo general. ____ *7*

Objetivos particulares. 7
1.5 Justificación. 8
1.6 Metodología. 8
1.7 Contenido de la tesis. 9
1.8 Marco Teórico. 10
1.8.1 Microcontrolador Arduino Mini 05. 10
1.8.2 Arduino IDE. 11
1.8.3 Sensor táctil capacitivo MPR121. 13
1.8.4 Bus de comunicación I^2C 14
1.8.5 Raspberry PI. 19
1.8.6 Sensor Infrarrojo. 24
1.8.7 Protocolos de comunicación infrarrojo. 24
1.8.8 Transformaciones afines. 26
1.8.9 Lenguaje de programación C Sharp. 32
1.9 Esquema general del proyecto. 33
***Capítulo 2 Marcador IR* 35**
2.1 Uso del Microcontrolador ATmega 328 mini. 36
2.1.1 Programación usando el módulo PL2303. 37
2.1.2 Librería Wire. 38
2.1.3 Librería IRremote. 39
2.2 Convertidor de Nivel Lógico Bidireccional. 40
2.3 Sensor táctil capacitivo MPR121. 42
2.4 Fuente regulada CD de 3.3 volts. 42
2.5 Touchpad. 43
2.6 Implementación del Marcador IR. 46
***Capítulo 3 Aplicación de escritorio* 47**
3.1 Programación dirigida por eventos. 49
3.2 Evento KeyDown. 49
3.3 Capturador de teclas (Keylogger). 50

3.4 Capturador de pantalla (ScreenShot) 51
3.5 Wi-Fi 53
3.6 Sockets 53
3.6.1 Creación 54
3.6.2 Conexión 55
3.6.3 Sockets Asíncronos 55
3.7 Cliente TCP/IP 55

Capítulo 4 Ras Pizarrón. ***57***
4.1 Servidor TCP/IP 59
4.2 Interacción Cliente-Servidor 60
4.3 Captura del plano de proyección 61
4.3.1 Ajuste de esquinas 62
4.3.2 Iniciar Ras Pizarrón 64
4.3.3 Búsqueda del pixel más brillante 65
4.4 Detección de códigos IR 68
4.5 Asignación de los códigos a gestos de ratón 70

Capítulo 5 Pruebas y Resultados. ***72***
5.1 Pruebas 72
5.2 Resultados 73

Capítulo 6 Conclusiones y trabajos futuros. ***79***
6.1 Conclusiones 79
6.2 Trabajos futuros 80

Anexo A. Código fuente marcador Ir ***81***

Anexo B. Código fuente para la aplicación de escritorio ***83***

Anexo C. Código fuente para la aplicación de la Raspberry Pi ***88***

Referencias ***97***

Lista de Figuras

Figura 1-1 Proyector de diapositivas con carrusel. 1
Figura 1-2 Retroproyector de laminillas transparentes. 2
Figura 1-3 Proyector de video. 3
Figura 1-4 Pizarra digital interactiva. 4
Figura 1-5 Teclado y ratón inalámbricos. 4
Figura 1-6 Mando a distancia con láser USB. 5
Figura 1-7 Mando a distancia infrarrojo. 6
Figura 1-8 Microcontrolador Arduino MINI 05. 10
Figura 1-9 Arduino IDE. 12
Figura 1-10 Sensor táctil capacitivo MPR121. 13
Figura 1-11 Reverso del sensor MPR121. 14
Figura 1-12 Conexión del Bus I2C con un maestro y dos esclavos 16
Figura 1-13 Condición de inicio del bus I2C. 17
Figura 1-14 Transmisión de la dirección al dispositivo esclavo. 17
Figura 1-15 Reconocimiento de la dirección por el dispositivo esclavo. 18
Figura 1-16 Transferencia de datos. 18
Figura 1-17 Reconocimiento de un byte recibido. 18
Figura 1-18 Fin de la transferencia de datos. 19
Figura 1-19 Transmisión de un byte. 19
Figura 1-20 Raspberry Pi modelo B+. 20
Figura 1-21 Raspbian Strech. 21
Figura 1-22 Modulo de cámara PI NOIR. 23
Figura 1-23 Sensor infrarrojo. 24
Figura 1-24 Traslación de una forma. a) Forma en posición original. b) Forma después de una traslación de (-15,5). 27
Figura 1-25 Rotación de una forma. a) Forma en posición original. b) Forma con una rotación de 45° y eje de rotación en (20,5). 29

Figura 1-26 Escalado de una forma. a) Forma en su posición original. b) Forma con escalado de (2/3,6/5). 31
Figura 2-1 Diagrama a bloques de la herramienta IR. 35
Figura 2-2 Conexión del Arduino MINI. 37
Figura 2-3 Conexión del módulo PL2303 y el Arduino MINI. 38
Figura 2-4 Circuito bidireccional de cambio de nivel. 41
Figura 2-5 Conexión del Convertidor de Nivel Lógico bidireccional. 41
Figura 2-6 Conexión del sensor táctil capacitivo a la fuente regulada. 43
Figura 2-7 Vista superior del Touchpad. 44
Figura 2-8 Vista inferior del Touchpad. 44
Figura 2-9 Implementación del Marcador IR. 46
Figura 3-3-1 Diagrama de flujo de la aplicación de escritorio. 47
Figura 3-2 Teclas monitoreadas por e Keylogger. 51
Figura 4-1 Diagrama general de Ras Pizarrón. 57
Figura 4-2 Interacción Cliente-Servidor. 60
Figura 4-3 Ajuste homográfico. 62
Figura 4-4 Búsqueda del pixel más brillante. 65
Figura 4-5 Nuevo pixel brillante. 66
Figura 4-6 Trazado de línea entre A y B. 67
Figura 4-7 Asignación de códigos NEC a los botones del Marcador IR. 68
Figura 4-8 Configuración del demonio lircd.conf. 69
Figura 4-9 Asignación de eventos de ratón para cada código NEC. 70
Figura 4-10 Configuración del demonio Lircmd.conf. 71
Figura 5-1 Prueba de visualización en un monitor. 72
Figura 5-2 Ras Pizarrón. 74
Figura 5-3 Ajuste de esquinas. 75
Figura 5-4 Comprobación de conectividad inalámbrica entre las aplicaciones. 76
Figura 5-5 Cambio de diapositiva. 76
Figura 5-6 Trazado de un polígono usando la herramienta. 77

Figura 5-7 Subrayado usando la herramienta. _____ *78*

Lista de Ecuaciones

1.1 Transformación afín _____ *27*
1.2 Transformación afín inversa _____ *27*
1.3 Traslación _____ *28*
1.4 Traslación inversa _____ *28*
1.5 Rotación _____ *29*
1.6 Rotación inversa _____ *30*
1.7 Escalado _____ *31*
1.8 Escaldo inverso _____ *31*

Lista de pseudocódigos

Pseudocódigo 2-1 Marcador IR _____ *45*
Pseudocódigo 3-1 Keylogger _____ *51*
Pseudocódigo 3-2 Capturador de pantalla _____ *52*
Pseudocódigo 3-3 Cliente TCP/IP _____ *56*
Pseudocódigo 4-1 Capturador del plano de proyección _____ *64*

Glosario de términos

Término	**Descripción**
.NET	Dominio de red
RAM	Memoria de Acceso Aleatorio
EEPROM	Memoria de solo lectura programable y borrable eléctricamente
E/S	Entrada y Salida
MBps	Megabytes por segundo
LED	Diodo Emisor de Luz
Ir	Infrarrojo
M	Metro
Nm	Nanómetros
I^2C	Inter-Circuitos-Integrados
A/D	Analógico/Digital
D/A	Digital/Analógico
SDA	Línea de Datos Serie
SCL	Señal de Sincronía
Ω	Ohm
mA	mili Amperes
V	Volt
RGB	Rojo, Verde, Azul (LED)
°C	Grados centígrados
mA	mili Amperes
PWM	Modulación de ancho de pulso
USB	Universal Serial Bus
IDE	Entorno de Desarrollo Integrado
CD	Corriente Directa

GPIO	Pines Entrada/Salida de Propósito General
TCP/IP	Protocolo de Control de Transmisión/Protocolo de Internet
Rx	Recepción
Tx	Transmisión
Pixel	Unidad mínima de una imagen digital
Wifi	Conexión inalámbrica de red
Demonio	Proceso informático que se ejecuta en segundo plano
Infragrama	Técnica de análisis infrarrojo de imágenes
Optoelectrónico	Dispositivo que combina la óptica y la electrónica

Capítulo 1

Introducción

La transmisión de conocimiento es vital para el desarrollo de la humanidad. Los seres humanos obtenemos gran parte de ese conocimiento mediante el lenguaje, pero se ha demostrado que, si se cuenta con un soporte visual a través de imágenes, la asimilación de las ideas sea mucho más sencilla, consiguiendo la permanecía de estas por mucho más tiempo.

El proyecto implementa uno de los cambios recientes en la forma tradicional de hacer presentaciones con diapositivas usando sistemas digitales de proyección, para ello vamos a hablar de los dos grupos de sistemas de proyección, los sistemas analógicos y los digitales (1).

1.1 Sistemas de proyección analógicos

Si bien, el proyecto se centra en los sistemas digitales de proyección, es interesante ver las características de las alternativas analógicas, con la finalidad de ampliar el contexto y conocer el punto de partida de este tipo de equipos.

Proyector de diapositivas

Figura 1-1 Proyector de diapositivas con carrusel.

Las diapositivas de este proyector permiten cierta portabilidad debido a su pequeño tamaño. Además, una vez situado en una posición fija solo es necesario configurarlo una vez. Por otro lado, no permite al ponente modificar las diapositivas en la mitad de la presentación y el carrete de diapositivas tienen una capacidad limitada que hace que se deban cambiar a menudo en mitad de la exposición perdiendo tiempo.

Retroproyector

Figura 1-2 Retroproyector de laminillas transparentes.

Este modelo de retroproyector permite que el ponente pueda señalar o incluso dibujar sobre la lámina, interactuando con ella. Como en el modelo anterior si se mantiene en una posición fija solo es necesario configurarlo una vez.

A pesar de ello, el sistema de cambio de lámina es completamente manual, lo que obliga a estar al lado del retroproyector y tener preparadas las láminas para evitar perder tiempo. Además, las láminas son de tamaño folio lo que hace que sean incomodas de transportar en grandes cantidades.

1.2 Sistemas de proyección digitales

Este tipo de sistemas, son más actuales y avanzados que sus homólogos analógicos, ya que permiten la proyección tanto de imágenes estáticas como de video.

Proyector de video

Figura 1-3 Proyector de video.

Los modelos avanzados permiten un control remoto básico para presentaciones y video. Suelen soportar entradas de video tanto analógicas como digitales, como por ejemplo la salida de una computadora. Permite un mayor grado de configuración de la imagen (saturación, luminosidad, contaste, etc.).

En cuanto a inconvenientes, se puede destacar que, si no se dispone de un equipo conectado y se conectan dispositivos diferentes cada vez, en ocasiones es necesario configurar cada vez la señal de video para que se visualice correctamente.

Pizarra interactiva

Figura 1-4 Pizarra digital interactiva.

La principal función de la pizarra es controlar la computadora mediante esta superficie con un bolígrafo, el dedo (en algunos modelos) u otros dispositivos como si se tratara de un ratón. Una mayor interactividad es lo que diferencia de una presentación digital tradicional (ordenador +proyector de video) (2).

1.3 Sistemas de control

Una vez asentado el tipo de sistema que vamos a utilizar (proyección de video procedente de una computadora) hay varias opciones para realizar el control de la presentación.

Teclado y ratón

Figura 1-5 Teclado y ratón inalámbricos.

El método de control por excelencia de las computadoras y responsable también de su popularización. Para avanzar y/o retroceder dentro de la presentación podemos presionar los botones del ratón, o las teclas de dirección del teclado.

Además, el puntero del ratón sirve para llamar la atención sobre alguna parte concreta y también permite seleccionar partes importantes de la diapositiva a través de los pinceles que los programas de visualización de diapositivas traen incorporados.

En cuanto a las ventajas, es interesante destacar su precio, su facilidad de uso y el control avanzado sobre el sistema y de las presentaciones. Por otra parte, es incómodo para utilizar de pie. Además, puede producirse una pérdida de conectividad si se trata de dispositivos inalámbricos.

Mando a distancia USB

Figura 1-6 Mando a distancia con láser USB.

El mando a distancia USB es un dispositivo que consta de dos partes, el mando propiamente y un receptor USB que se debe introducir en la computadora. Funciona como mando a distancia y suele tener incorporado un láser para poder señalar sobre la pantalla de proyección y así hacer hincapié sobre

el contenido. Normalmente sus funciones son las de avanzar/retroceder la presentación, cerrar la aplicación y subir o bajar volumen.

Suele requerir una configuración sencilla, tiene un reducido tamaño y bajo costo. Por otra parte, presenta perdida de conectividad frecuente dependiendo del modelo y un control muy básico.

Mando a distancia Infrarrojo

Figura 1-7 Mando a distancia infrarrojo.

Se trata de un dispositivo de control que emplea un LED infrarrojo para enviar una señal, dicha señal es recibida con un detector infrarrojo y esta puede ser utilizada para accionar algún tipo de mecanismo o simplemente para cambiar alguna característica en los dispositivos multimedia.

Como este tipo de mandos utilizan luz como medio de transmisión, es necesario tener una línea de visión directa con el receptor, de lo contrario, el haz puede no ser recibido por el receptor. Dicha señal de luz tiene una longitud de onda típica de 940 nm.

Al igual que cualquier tipo de comunicación, esta también sigue un conjunto de normas y de estándares los cuales deben ser respetados para poder cumplir con la transmisión de las señales.

1.4 Objetivo.

Objetivo general.

Diseñar e implementar una herramienta que permita asemejar sus funciones a las de una pizarra interactiva, con el fin de permitir al ponente dibujar, subrayar e interactuar con las diapositivas, en las presentaciones digitales de cualquier tipo, con un costo mucho menor que el de la pizarra y sin las restricciones comerciales que a menudo se imponen con estos dispositivos.

Objetivos particulares.

- ✓ Diseñar una herramienta remota con las características de un mando infrarrojo a distancia, pero con el tamaño de un marcador para pizarrón.

- ✓ Implementar una aplicación de escritorio, capaz de monitorear los eventos generados por el usuario, sobre la presentación de diapositivas.

- ✓ Implementar una aplicación para Raspberry Pi, capaz de monitorear mediante cámara la ubicación del Marcador IR.

- ✓ Establecer una comunicación inalámbrica mediante Wifi entre ambas aplicaciones.

- ✓ Aplicar las configuraciones necesarias al Marcador IR, con la finalidad de que su comportamiento, asemeje al de un ratón de computadora.

1.5 Justificación.

El aprendizaje es un proceso fundamental en la vida cotidiana, con él se adquieren nuevas habilidades, destrezas y conocimientos. Por esta razón, se están desarrollando nuevas tecnologías que facilitan este proceso, tal es el caso de las presentaciones digitales con imágenes o incluso con video.

En el mercado son pocos los dispositivos que permiten al usuario interactuar con la presentación de diapositivas, y los pocos existentes tienen costos bastante elevados, por tal motivo este proyecto plantea diseñar una herramienta similar a una pizarra electrónica, pero con un costo mucho menor.

1.6 Metodología.

Previo al desarrollo del proyecto, se realiza una investigación bibliográfica y en línea, acerca de los avances tecnológicos disponibles para mejorar la interacción en las ponencias, a través de presentaciones con dispositivos de proyección digital. En base a esto, se plantea claramente el alcance del proyecto, así como sus objetivos.

Luego, se realiza una investigación de cada uno de los conceptos teóricos elementales, tales como; la transformación de imágenes afines y la homografía, así como de las tecnologías y los elementos necesarios para la implementación del software y el hardware.

Posteriormente, se buscan los dispositivos electrónicos disponibles en el mercado, que cumplen con los requerimientos de hardware y software necesarios para llevar a cabo las tareas requeridas por la herramienta. Estos dispositivos son: Arduino y Raspberry Pi en hardware y en software se aborda la programación en .Net y Arduino IDE.

Se desarrollarán las aplicaciones, tanto la aplicación de escritorio, como la aplicación para la Raspberry Pi en .Net, donde previamente se indagó en algunos de los recursos necesarios, tales como; Keylogger y los Sockets.

1.7 Contenido de la tesis.

En el Capítulo 1 se hace una pequeña introducción a este proyecto, se muestran los antecedentes de las herramientas de proyección digitales. Además, se menciona el objetivo del proyecto y se justifica la elección del mismo, después se describen las herramientas necesarias en el marco teórico, así mismo se muestra el esquema generalizado del proyecto junto con las características de cada uno de sus elementos.

En el Capítulo 2 se aborda el diseño del Marcador IR, incluyendo la descripción de cada dispositivo electrónico necesario para su construcción.

En el Capítulo 3 se describe a detalle la construcción de la Aplicación de Escritorio, con cada uno de los recursos computacionales que la hacen ser indispensable para este proyecto.

En el Capítulo 4 se aborda la Aplicación para la Raspberry Pi, la cual recibe el nombre de Ras Pizarrón, donde se describen los elementos que la integran y del proceso que sigue para efectuar la detección del Marcador IR.

En el Capítulo 5 se presentan los resultados y algunas pruebas hechas a la herramienta de proyección.

En el Capítulo 6 aparecen las conclusiones y observaciones obtenidas de la implementación de este proyecto, así como los trabajos futuros.

1.8 Marco Teórico.

1.8.1 Microcontrolador Arduino Mini 05.

Se trata de una pequeña tarjeta de desarrollo Arduino, basada en un microcontrolador ATmega 328, por su tamaño, es ideal para trabajar en proyectos con poco espacio. Sus características hacen que este dispositivo pueda ser utilizado en una gran variedad de aplicaciones, por ejemplo: teléfonos inteligentes, accesorios de audio, periféricos para juegos de video, dispositivos médicos avanzados, entre otras (3).

El dispositivo cuenta con puertos de entrada/salida, módulos PWM, comparadores, temporizadores y periféricos de comunicación avanzados.

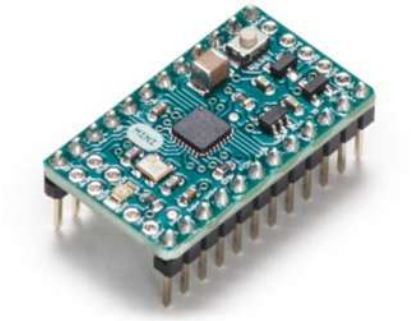


Figura 1-8 Microcontrolador Arduino MINI 05.

Especificaciones técnicas.

En la siguiente tabla muestra las principales características de este microcontrolador (3).

Microcontrolador	**ATmega328**
Voltaje de operación	5V
Voltaje de alimentación	7-9 V
Pines digitales E/S	14 (6 cuentan con salida PWM)
Pines analógicos de entrada	8

Corrientes DC en los pines E/S.	40 mA
Memoria Flash	32 KB (donde 2 KB son usados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

Tabla 1-1 Especificaciones técnicas del microcontrolador.

Este microcontrolador cuenta con el bus de comunicación I^2C, lo que lo hace ideal para trabajar en modo maestro o esclavo, en conjunto con otro microcontrolador o algún sensor en específico.

El dispositivo carece de una interfaz USB, por ello, es necesario hacer uso de un adaptador USB serial para poder programarlo, en nuestro caso usaremos el adaptador con matrícula PL0323.

Para poder utilizar el microcontrolador y disponer de todas sus funciones, es necesario del uso de un software, con el cual, sea posible escribir las instrucciones a realizar mediante código fuente, compilarlo y posteriormente programarlo.

1.8.2 Arduino IDE.

El software Arduino IDE (Entorno de Desarrollo Integrado por sus siglas en inglés), está basado en Processing (4) y lenguaje de programación Wiring (5), así como, en el cargador de arranque (bootloader) que es ejecutado en la placa.

Arduino se enfoca en acercar y facilitar el uso de la electrónica y sobre todo la programación de sistemas embebidos en proyectos multidisciplinarios, brindando bibliotecas disponibles para una amplia variedad de sensores. Toda la plataforma, tanto para sus componentes de hardware como de software, son

liberados con licencia de código abierto, permitiendo libertad de acceso para cualquier usuario.

En otras palabras, Arduino IDE es un entorno que nos permite comunicar el microcontrolador Arduino con la computadora y así poder integrarle cualquier programa escrito en C o cualquier lenguaje de programación que derive de C. Ese programa podrá interactuar con cada salida o entrada de nuestro Arduino.

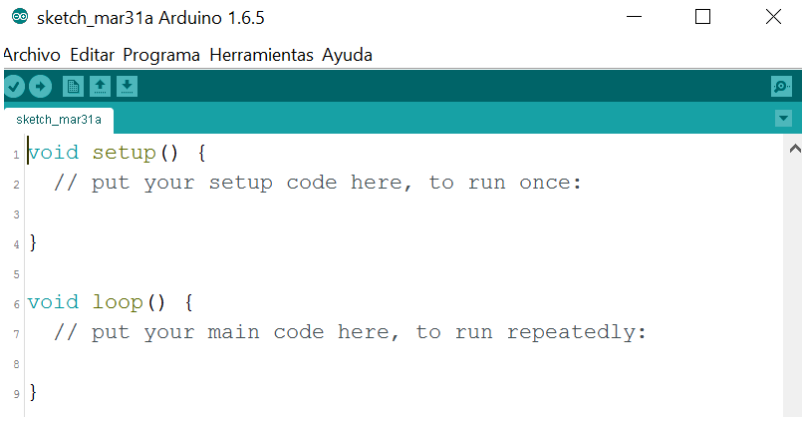


Figura 1-9 Arduino IDE.

En la Figura 1-9 se muestra el entorno de programación Arduino IDE, donde se observan dos principales bloques de código, cada una a la espera de las instrucciones correctas.

Void setup: en este bloque de código van todas las definiciones y declaraciones indispensables para el funcionamiento del programa.

Void loop: aquí se colocan las instrucciones que ejecutara el programa de manera repetitiva.

1.8.3 Sensor táctil capacitivo MPR121.

Se trata de un dispositivo electrónico capáz de detectar los cambios capacitivos producidos en la superficie de conectada a los electrodos. El módulo MPR121 puede controlar 12 electrodos, cada electrodo se puede comportar como un sensor táctil capacitivo (6).

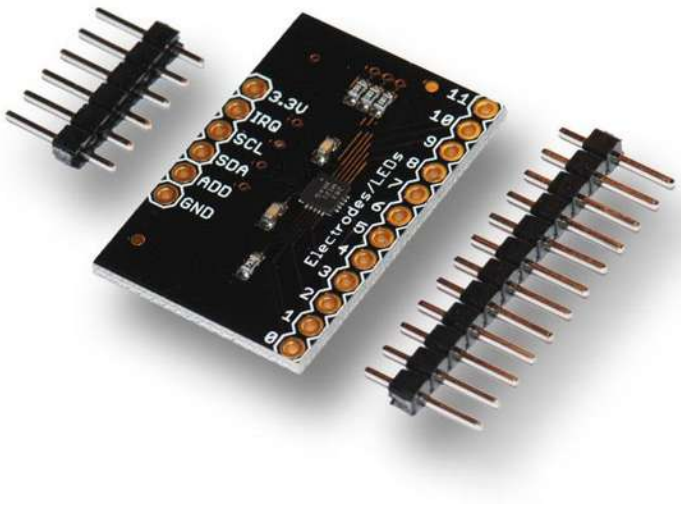


Figura 1-10 Sensor táctil capacitivo MPR121.

El electrodo puede ser cualquier superficie conductora, como metal, papel metálico, cobre, etc. Para activar el electrodo no necesariamente tiene que haber contacto, la sensibilidad depende de cómo se calibre el sensor. La sensibilidad puede ser ajustada en un rango de X a Y donde Y detecta a una determinada distancia (sin haber tocado el electrodo) el acercamiento del dedo, mediante la elección de un umbral se determinará "toque o activación".

En caso no se utilicen todos los electrodos, también se puede configurar como salidas para activar LED'S.

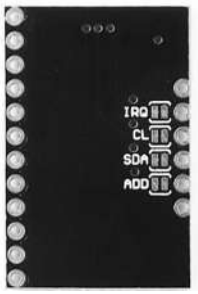


Figura 1-11 Reverso del sensor MPR121.

En el reverso del módulo MPR121, encontramos 4 puentes, los cuales hay que romper si no se desea usar las resistencias pull-up que tiene el módulo. También se pueden romper los puentes de ADD para cambiar la dirección en caso se necesite conectar otro MPR121 al bus I2C (6).

La forma en cómo se comunica este dispositivo es mediante interfaz I^2C, por lo que solo utiliza dos pines y se puede implementar con cualquier microcontrolador.

1.8.4 Bus de comunicación I^2C

El bus de comunicación I^2C (Inter Integrated Circuit), es un protocolo de comunicación diseñado por Philips Semiconductors (ahora NXP Semiconductors), que se utiliza para la comunicación entre circuitos integrados (7).

Actualmente, este protocolo de comunicación se implementa en más de 1000 circuitos integrados fabricados por más de 50 empresas (7), teniendo una gran aceptación en el mercado por lo que los principales fabricantes de dispositivos semiconductores incluyen al menos un bus I^2C que permita la interacción con otros dispositivos. Algunos dispositivos que incorporan este bus de comunicación son los siguientes:

- ✓ Actuadores (servos, válvulas, etc.)
- ✓ Convertidores A/D y D/A
- ✓ Sensores (temperatura, inclinación, velocidad, campo magnético, etc.)
- ✓ Memoria EEPROM
- ✓ Reloj/calendario de tiempo real (RTC)
- ✓ LCD con driver
- ✓ Microcontroladores

Características

El bus I^2C se caracteriza por que la comunicación se realiza a través de dos líneas de señal, además del común o tierra. La metodología de comunicación es en serie y síncrona. Una de las señales se utiliza para enviar la información mientras que la otra controla la sincronización del reloj. En seguida se describen cada una de estas señales (6).

- ✓ **SDA** (*System Data*) – línea por donde se mueven los datos de un dispositivo a otro.
- ✓ **SCL** (*System Clock*) – línea de reloj que sincroniza el sistema.
- ✓ **GND** (*Masa*) – línea común de conexión entre todos los dispositivos conectados al bus.

Para realizar la comunicación se necesita de al menos un dispositivo configurado como maestro, el cual tiene la iniciativa en la transferencia de datos, decide a que dispositivos esclavos enviar la información y cuando finalizar la comunicación. Los esclavos conectados al bus, tienen la función de recibir los datos enviados por el maestro. Cada dispositivo conectado al bus tiene una dirección asignada por el usuario. Cuando el maestro inicia una comunicación primero envía la dirección del dispositivo con el que se quiere comunicar y los esclavos comprueban si la dirección concuerda con la suya, en caso de que la dirección coincida con un dispositivo, este enviará una señal de conformidad

indicando al maestro que está listo para recibir los datos. Los demás dispositivos al no corresponder a esa dirección, ignoraran la solicitud y se pondrán todos a la espera de que el maestro inicie una nueva secuencia de comunicación (6).

Las dos líneas del bus son del tipo drenador abierto, es decir un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo. Se deben polarizar en estado alto por medio de resistencias pull up que defina una estructura de bus que permita conectar en paralelo múltiples entradas y salidas.

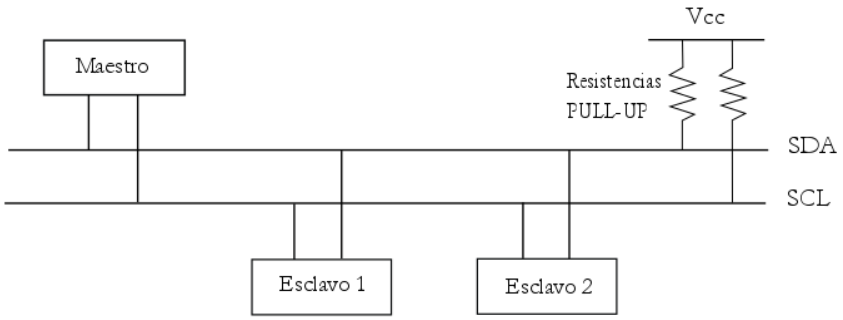


Figura 1-12 Conexión del Bus I2C con un maestro y dos esclavos

El bus de comunicación I^2C, como todo caso, presenta ventajas y desventajas frente a otros protocolos de comunicación, por lo que es importante considerarlas antes de desarrollar una aplicación.

Ventajas

- ✓ Pocos cables de interconexión
- ✓ Conexión de dispositivos a distancia
- ✓ Componentes con encapsulado reducido

Desventajas

- ✓ Velocidad inferior en comparación a otros protocolos

- ✓ Disponibilidad de circuitos que soporten el bus

Protocolo de programación

El protocolo de programación que se describirá a continuación sólo muestra los pasos para una configuración donde se tiene un solo maestro y varios esclavos, es decir unidireccional. Para más información acerca de este protocolo, se pueden consultar el manual señalado en la bibliografía (7).

La transmisión de datos se inicia con la condición de inicio (START) de transferencia de datos. El dispositivo maestro deberá poner a nivel bajo la línea de datos (SDA) y dejando a nivel alto la línea de reloj (SCL).

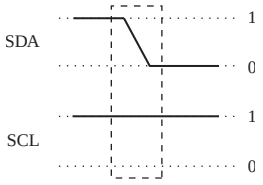


Figura 1-13 Condición de inicio del bus I2C.

Al iniciar la transmisión, el maestro envía la dirección del esclavo con el que desea establecer la comunicación. La dirección contiene siete bits que componen la dirección del dispositivo al que se desea enviar la información. Tras la dirección se adjunta un octavo bit que corresponde la operación que se va a realizar (0 = escritura, 1 = lectura).

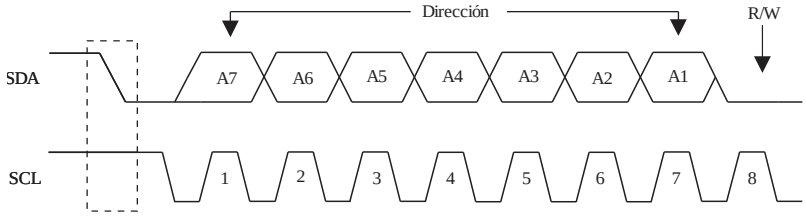


Figura 1-14 Transmisión de la dirección al dispositivo esclavo.

Una vez que el maestro envía la dirección, el esclavo genera un bit de reconocimiento (ACK) ubicado después del octavo bit que ha enviado el maestro. Si no se genera este bit de reconocimiento, la comunicación se interrumpe generando una señal de STOP.

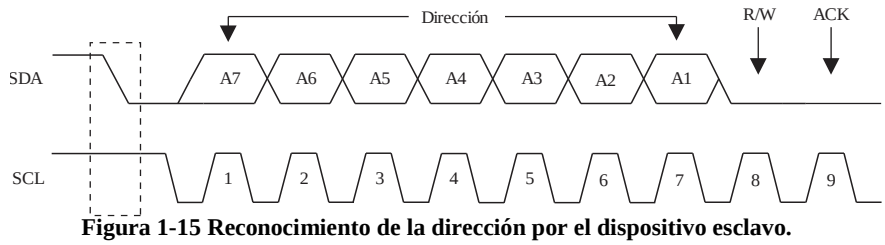


Figura 1-15 Reconocimiento de la dirección por el dispositivo esclavo.

El bit de reconocimiento ACK, le permite al maestro saber si el esclavo reconoció la solicitud para iniciar el intercambio de información. En ese momento el maestro comienza a transmitir los datos.

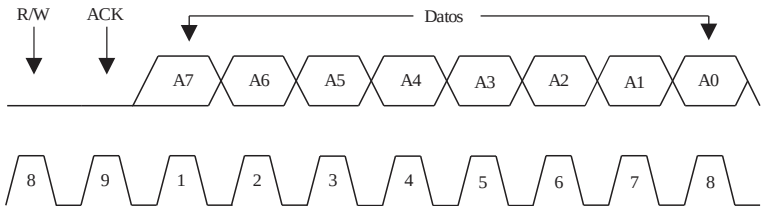


Figura 1-16 Transferencia de datos.

Al enviarse el primer dato, el esclavo deberá generar un bit de reconocimiento ACK después del octavo bit. Por cada byte enviado se debe de poner el noveno bit a nivel lógico bajo (ACK=0) para continuar enviando información, si no se realiza esto, la comunicación se interrumpe y se genera una señal de STOP.

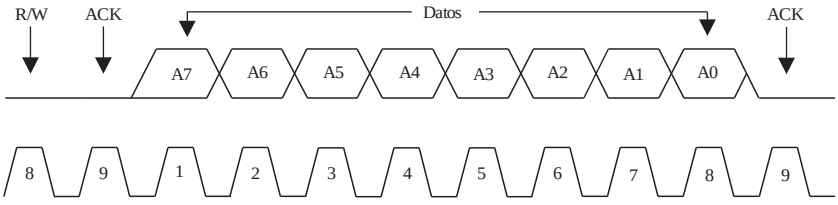


Figura 1-17 Reconocimiento de un byte recibido.

La condición de fin de transferencia se produce cuando un dispositivo maestro pone en estado alto la línea de datos (SDA), de igual manera la línea de reloj (SCL) también se deja en estado alto (7).

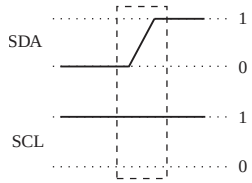


Figura 1-18 Fin de la transferencia de datos.

Básicamente este es el protocolo que se sigue en el bus de comunicación I^2C para mandar información de un byte, el cual se resume en la figura 3.18.

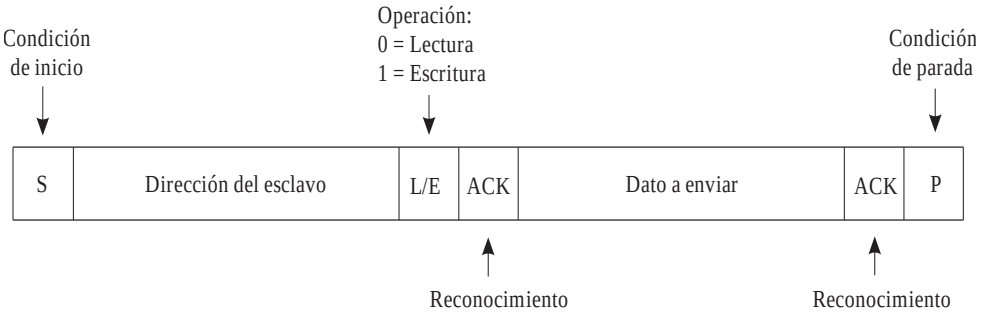


Figura 1-19 Transmisión de un byte.

1.8.5 Raspberry PI.

Raspberry PI, es un dispositivo moderno, basado en una placa de computadora de bajo costo, en otras palabras, se trata de una computadora de tamaño reducido, del tamaño de una tarjeta de crédito, desarrollado en el Reino Unido por la Fundación Raspberry PI en el 2011, con el objetivo de estimular la enseñanza de la informática en las escuelas de nivel básico, aunque no empezó su comercialización hasta el año 2012.

El concepto, es el de una computadora desnuda básica, formado por una placa que soporta varios componentes necesarios en una computadora común y es capáz de comportarse como tal (8).

Figura 1-20 Raspberry Pi modelo B+.

Raspberry Pi Modelo B+, es una mini computadora con todas las cualidades de una computadora normal, con los elementos básicos de hardware y software.

Cuenta con:

- ✓ Procesador BCM 2835.
- ✓ 512 MB de RAM.
- ✓ incluye reguladores lineales para bajar el consumo a 0.5 watt.
- ✓ Ranura especial para módulo de Cámara.
- ✓ El cabezal GPIO (Pines de entrada y salida para propósito general), cuenta con cuarenta pines, los cuales están a disposición del usuario.

Por otro lado, en el dispositivo nos encontramos con 4 puertos USB 2.0 y también cabe mencionar que cuenta con ranura de microSD de clic para ponerla

y quitarla de forma más fácil, dicha microSD es utilizada para montar la imagen ISO del sistema operativo que ejecutara el dispositivo (9).

Existen diferentes sistemas operativos que pueden ser utilizados por la Raspberry, aunque la mayoría de sus aplicaciones se centra mayormente sistemas operativos GNU/Linux, su versión original es denominada Raspbian, la cual es una versión adaptada de Debian, lo que convierte a este dispositivo en una herramienta de software libre.

Raspbian Stretch.

Se trata de la última versión de software libre disponible para Raspberry PI, basado en entorno Linux, la cual asemeja todas las características de Debian (10), solo que adaptadas a Raspberry. Los lanzamientos de Debian, llevan el nombre de personajes de la trilogía de Toy Story de Disney Pixar®. Stretch es un pulpo.

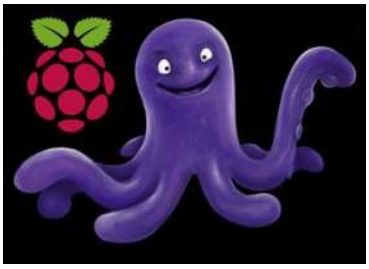


Figura 1-21 Raspbian Strech.

Este software está disponible para ser descargado por cualquier usuario desde el sitio web de Raspberry. Está disponible en dos versiones, una que incluye interfaz de escritorio y una versión mínima, la cual solo contiene el entorno terminal.

El software tiene que ser montado en una tarjeta SD, con capacidad mínima de 4 Gb, los pasos a seguir para la instalación se muestran en la referencia (9).

LIRC.

LIRC (Linux Infrared Remote Control), es una paquetería de código abierto disponible para Linux, que permite enviar y recibir comandos de señales infrarrojos, usadas en la mayoría de los mandos a distancia (11).

Una de las partes importantes, es que cuenta con demonios capaces de realizar acciones específicas sobre las señales infrarrojas.

- ✓ ***Demonio lircd,*** el demonio que decodifica las señales Ir recibidas por el controlador y proporciona la información por un socket.

- ✓ ***Demonio lircmd,*** este demonio se conecta a *lircd* y traduce las señales Ir decodificadas a movimientos y gestos de ratón. Por lo tanto, se puede configurar un programa para usar el mando a distancia como un dispositivo de entrada.

Las aplicaciones que se ejecutan en espacio de usuario permiten controlar la computadora con el mando. Se pueden enviar eventos X a los programas, arrancar programas, y mucho más sólo con una pulsación de un botón. Algunas posibilidades que esto ofrece son: ratón sin cables, control remoto para un sintonizador de TV, apagado a distancia de la computadora, programación de un grabador de vídeo, etc.

Sensor de cámara PI CÁMARA.

Una de las ventajas que ofrece Raspberry, es la comodidad para trabajar con diferentes dispositivos, tal es el caso del módulo de cámara Pi Cámara, el cual

está diseñado especialmente para trabajar con la Raspberry, principalmente es usado en aplicaciones de reconocimiento de formas y patrones, en áreas como; Infragrama (para medir la salud de las plantas), astrofotografía y fotografía de la fauna nocturna (12).

Figura 1-22 Modulo de cámara PI NOIR.

Este módulo de cámara, es compatible con cualquier Raspberry Pi (1,2,3, Zero), el cual puede ser utilizado para crear fotografías y vídeo de muy buena calidad. El módulo Pi Camera (sin infrarrojos) es igual que el módulo de cámara estándar, pero sin filtro de infrarrojos. Por tanto, es excelente para fotografía y vídeo en la oscuridad o crear "Infragramas" .

El módulo de cámara, utiliza el sensor de imagen IMX219PQ de Sony que ofrece imágenes de vídeo de alta velocidad y alta sensibilidad.

Pi Cámara ofrece contaminación de imagen reducida. Dispone también de funciones de control automático como el control de exposición, el balance de blancos y la detección de iluminación.

La instalación del módulo de cámara, únicamente utiliza un cable plano fijado a las ranuras de la Raspberry, directamente en el puerto de interfaz serie de su cámara Pi (CSI).

La cámara es sensible a la radiación por infrarrojos de onda corta (aprox. 880 nm) producida por dispositivos LED de emisión de infrarrojos. No es sensible a la radiación por infrarrojos de onda larga (calor), por lo que se requiere iluminación LED IR para poder ver en la oscuridad (13).

1.8.6 Sensor Infrarrojo.

El sensor infrarrojo, es un dispositivo optoelectrónico capaz de medir la radiación electromagnética infrarroja. Todos los cuerpos emiten una cierta cantidad de radiación, esta resulta invisible para los ojos humanos, pero no para este dispositivo, ya que se encuentra en el rango del espectro justo por debajo de la luz visible.

Figura 1-23 Sensor infrarrojo.

1.8.7 Protocolos de comunicación infrarrojo.

Philips RC5

Se trata del protocolo de comunicaciones infrarrojas diseñado por Philips en 1980, se basa en la codificación bifase, la cual, es un método de codificación donde por cada periodo de bit de datos, existe una transición entre dos niveles

de señal. Este tipo de codificación también es conocida como codificación Manchester (14).

Por otro lado, los mensajes de este protocolo, están divididos por partes, el comando y la dirección del dispositivo infrarrojo, donde este último está compuesto por 5 bits y el de comando por 6 bits. En su totalidad el bloque contiene 14 bits, de los cuales los primeros 2 son los bits de inicio y el tercero es el bit de cambio. Y la duración de envió de los bits dura aproximadamente 25 ms.

Sony SIRC

Este protocolo presenta tres versiones diferentes; 12, 15 y 20 bits, en las tres versiones la estructura es la misma, debido a que este cuenta con bits de inicio, bits de comando, y por último los bits de dirección del dispositivo. En la versión de 20 bits permite transmitir 8 bits para el comando del dispositivo. Cabe mencionar que la frecuencia de este protocolo es de 40 KHz (15).

NEC

Este protocolo trabaja a 38 kHz de frecuencia y su principal característica es que envía doble vez el comando, una vez de manera normal y la segunda de manera negada, con lo cual se hace más difícil transmitir datos erróneos. Los bits que se transmiten con este protocolo vienen acomodados casi de la misma forma que en los protocolos anteriores, primero los bits de inicio, luego los bits de dirección del dispositivo solo que esta vez se repiten dos veces y luego los bits de comando a ejecutar.

Este protocolo es utilizado por la librería IRemote que provee Arduino IDE, lo cual permite tener este protocolo disponible para la transmisión de datos vía infrarrojo.

1.8.8 Transformaciones afines.

Las imágenes digitales, así como las secuencias de imágenes digitales provenientes del mundo continuo, nos brindan un amplio panorama de lo que ocurre con la mayoría de los fenómenos físicos. Éstas se obtienen a partir de imágenes análogas mediante muestreo y cuantificación, procesos que dependen de los dispositivos de captura de imagen a los cuales les llamamos cámaras digitales.

La técnica de transformación afín, se utiliza típicamente para corregir las distorsiones o deformaciones geométricas que se producen con ángulos de cámara no ideales (16). Por ejemplo, las imágenes de satélite utilizan transformaciones afines para corregir la distorsión de la lente gran angular, la costura panorámica y el registro de imágenes. La transformación y fusión de las imágenes a un sistema de coordenadas grande y plano es deseable para eliminar la distorsión.

Las transformaciones Geométricas modifican la relación espacial entre píxeles. En términos del procesamiento de imágenes digitales una transformación geométrica consiste de dos operaciones básicas:

1. Una transformación espacial que define la reubicación de los píxeles en el plano imagen.

2. Interpolación de los niveles de grises, los cuales tienen que ver con la asignación de los valores de intensidad de los píxeles en la imagen transformada.

En términos Matemáticos, las transformaciones afines son las más usadas en imágenes digitales 2D por su representación y manejo matricial (17).

Una Transformación afín es aquella transformación en la que las coordenadas (x' , y'), del punto imagen son expresadas linealmente en términos

de las del punto original (x, y). Es decir, la transformación viene dada por las ecuaciones:

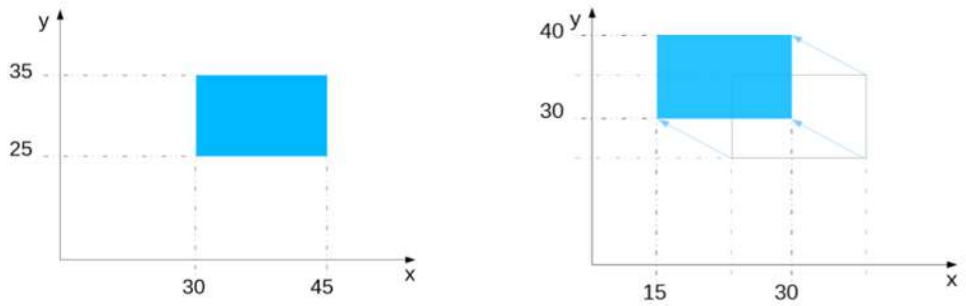
$$x' = ax + by + m$$
$$y' = cx + dy + n \qquad \textbf{(1.1)}$$

Cuando $m = n = 0$ las ecuaciones anteriores se convierten en el prototipo de una transformación lineal multiplicación por una matriz a la izquierda, esto es:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \textbf{(1.2)}$$

Traslación

Este tipo de transformación, es utilizada para cambiar la posición actual de un de un objeto a una nueva posición en el espacio de coordenadas, preservando la forma, el tamaño y la orientación (18). Es decir, mover la posición original del objeto, a lo largo de una trayectoria en línea recta hacia una nueva ubicación.



a) **b)**

Figura 1-24 Traslación de una forma. a) Forma en posición original. b) Forma después de una traslación de (-15,5).

Generalmente, se usan coordenadas homogéneas para representar la traslación mediante una matriz y poder así expresarla como una transformación lineal sobre un espacio de dimensión superior.

En esas condiciones, la forma de la Figura 1-24, sufre una traslación desde sus coordenadas originales (x, y), hasta unas nuevas coordenadas (x', y'), en otras palabras, se añaden distancias de traslación a las coordenadas originales.

Matemáticamente, esta transformación puede ser representada por una matriz como:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(1.3)}$$

Esta transformación, también permite ser utilizada en el caso contrario, es decir, cuando se desea pasar de las coordenadas (x', y') a (x, y). Para lo anterior, únicamente es obtener la inversa de la matriz anterior, quedando de la siguiente manera:

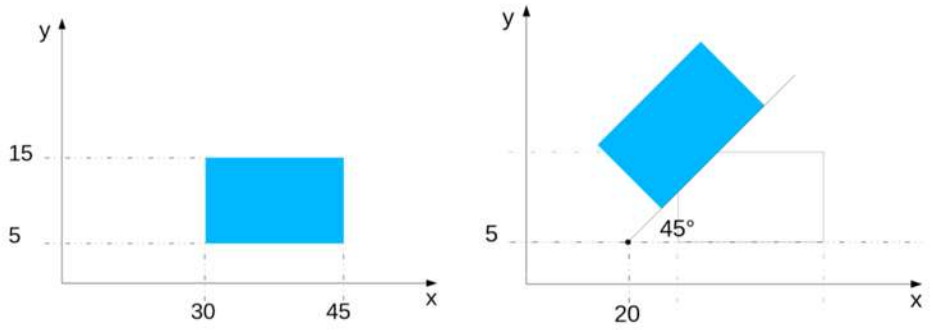
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \textbf{(1.4)}$$

El uso de esta transformación, no se limita únicamente a planos en dos dimensiones, sino que también puede ser utilizada en traslaciones en plano 3D.

Rotación

Esta transformación, efectúa un cambio de orientación sobre un punto o un objeto, es decir, hace girar el objeto una cantidad específica, dada por la magnitud de un ángulo, tomando como referencia un eje de rotación (19). El eje de rotación esta dado por un punto coordenado. La clasificación de los ángulos se da de dos formas:

- Ángulos positivos: rotan en sentido antihorario.
- Ángulos negativos: rotan en sentido horario.



a) **b)**

Figura 1-25 Rotación de una forma. a) Forma en posición original. b) Forma con una rotación de 45° y eje de rotación en (20,5).

Como se observa en la Figura 1-25, la forma sufre una rotación, por ende, sus coordenadas originales (x, y), también sufren un cambio y se convierten en unas nuevas coordenadas (x', y'). Este cambio en las coordenadas puede ser expresado en la siguiente forma matricial:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -sen\,\theta & 0 \\ \text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(1.5)}$$

Esta transformación solo es posible para ángulos negativos, es decir, únicamente para cuando se desea pasar de las coordenadas originales (x,y) a (x',y').

Para el caso contrario, es decir, cuando se desea pasar de las coordenadas (x' , y') a (x , y), con ángulo de rotación positivo, existe la posibilidad de obtener la matriz inversa de la expresión, dando el siguiente resultado:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & sen\,\theta & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \textbf{(1.6)}$$

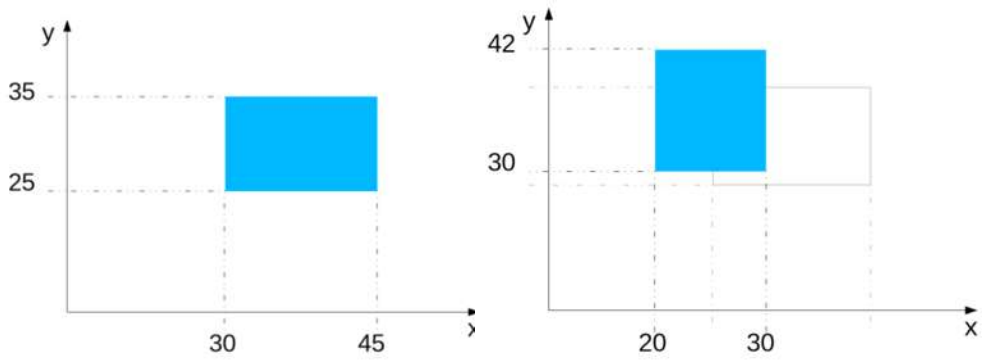
Este tipo de transformación, además de ser utilizada en planos de únicamente dos dimensiones, también permite su uso en planos 3D.

Escalado

El escalado de una forma, es una alteración geométrica en el tamaño del objeto, esta transformación, es efectuada cuando las posiciones de la forma son multiplicadas, por factores de escala en cada eje de coordenadas (19).

Estos factores de escala pueden clasificarse según la alteración que producen en la forma, quedando de la siguiente manera:

- ✓ Valores inferiores a 1: reducen el tamaño de la forma.
- ✓ Valores superiores a 1: aumentan el tamaño de la forma.



a) **b)**

Figura 1-26 Escalado de una forma. a) Forma en su posición original. b) Forma con escalado de (2/3,6/5).

Las coordenadas (x , y), de la Figura 1-26, sufren una alteración por motivo del escalado, es por ello, que las nuevas coordenadas (x' , y'), presentan un factor de escalamiento, el cual puede ser representado de manera matricial, como sigue:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(1.7)}$$

De igual manera, esta transformación puede ser expresada de manera inversa, es decir, pasar de coordenadas (x' , y') a (x , y), esto se obtiene únicamente con la inversa de la matriz anterior, quedando de la siguiente manera:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{1}/_{S_x} & 0 & 0 \\ 0 & {}^{1}/_{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \textbf{(1.8)}$$

No solo es posible usar escalado en 2D, también es posible utilizarlo en 3D.

1.8.9 Lenguaje de programación C Sharp.

C Sharp, o mejor conocido como C#, es un lenguaje de programación orientado a objetos desarrollado por Microsoft®, como parte de su plataforma .NET, es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Su nombre fue inspirado por el signo #, debido a que este se compone de cuatro signos "+" pegados (20).

Su sintaxis se deriva de los lenguajes C y C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C Sharp forma parte de la plataforma .NET, esta es una API (Interfaz de Programación de Aplicaciones), es un lenguaje de programación independiente diseñado para generar aplicaciones sobre dicha plataforma. Por otro lado, existe un compilador implementado que provee Mono-DotGNU (21), el cual genera aplicaciones para diversas plataformas, entre ellas GNU/Linux.

1.9 Esquema general del proyecto.

El proyecto está compuesto por tres elementos principales que en conjunto hacen funcionar de manera óptima la herramienta, estos elementos son:

- ✓ Un marcador o lápiz infrarrojo.
- ✓ Una aplicación de escritorio.
- ✓ Una aplicación para Raspberry Pi.

Cada uno de los elementos cumplen con tareas específicas y reciben características totalmente diferentes una de la otra, pero forzosamente todas deben estar en operación simultánea para dar vida a la herramienta.

Figura 1-9 Esquema general del proyecto.

En la Figura 2-1, se aprecian gráficamente cada uno de los elementos y la forma de comunicación entre ellos. A continuación, se describen de manera general las tareas y las características que deben cumplir cada uno.

Marcador IR:

- ✓ Establecerá la comunicación con la Raspberry Pi por medio de infrarrojo.
- ✓ Incorporará los gestos de un ratón de computadora.
- ✓ Contará con un tamaño semejante al de un marcador para pizarrón.
- ✓ Implementará el bus de comunicación I^2C entre sus componentes.

Aplicación de escritorio:

- ✓ Actuará en segundo plano, sin interrumpir las tareas del usuario.
- ✓ Interceptará únicamente las teclas de dirección y la tecla enter.
- ✓ Efectuará una captura de pantalla de la computadora personal del usuario.
- ✓ Activará el modelo de comunicación TCP/IP cliente y enviar la captura de pantalla a una ubicación de red.

Aplicación Ras Pizarrón:

- ✓ Activará el modelo de comunicación TCP/IP servidor.
- ✓ Obtendrá la imagen de la captura de pantalla ubicada en la dirección de red y la pondrá en el plano de proyección.
- ✓ Tomará fotografías del plano de proyección y procesará cada una de ellas para determinar la posición del Marcador IR.
- ✓ Capturará los códigos infrarrojos enviados por el Marcador IR y transformarlos a eventos de ratón.

Posteriormente, se dedica un capítulo completo para describir el diseño de cada uno de los elementos mencionados con anterioridad.

Capítulo 2
Marcador IR

En este capítulo se aborda el desarrollo necesario para llevar a cabo la construcción del Marcador IR, el cual tiene como función brindarle al usuario la posibilidad de interactuar con las presentaciones de diapositivas, brindándole la posibilidad de contar con un marcador o lápiz digital y un ratón inalámbrico.

Esta herramienta, hace uso de diferentes dispositivos electrónicos para poder funcionar correctamente, estos son:

- ✓ Microcontrolador Arduino Mini ATmega328.
- ✓ Sensor táctil capacitivo MPR121.
- ✓ Convertidor de Nivel Lógico Bidireccional (BD-LLC).
- ✓ Fuente regulada de 5v a 3.3v.

En la Figura 2-1, se muestra el diagrama con cada uno de los bloques de los dispositivos electrónicos que integran herramienta, también se muestra la manera de conexión entre ellos.

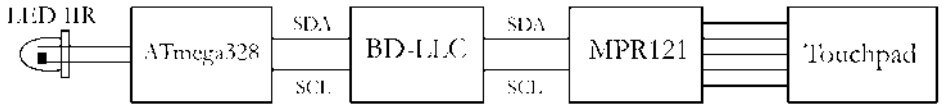


Figura 2-1 Diagrama a bloques de la herramienta IR.

La herramienta cuenta con el tamaño de un marcador para pizarrón convencional e integra las funciones de un mando a distancia infrarrojo, la única diferencia es que incluye un sensor táctil capacitivo, lo cual hace que se dejen de lado los típicos botones que se incluyen en los mandos a distancia comunes, en su lugar se hace uso de un Touchpad que responde a las pulsaciones táctiles, permitiendo al usuario utilizar sus dedos para operar el marcador.

La interacción entre los dispositivos electrónicos que integran esta herramienta, se lleva a cabo mediante el bus de comunicación I^2C, donde el dispositivo que funciona como maestro es el Microcontrolador ATmega 328 y el dispositivo esclavo es el sensor táctil capacitivo MPR121.

La fuente regulada se utiliza para bajar el voltaje de alimentación de 5v a 3.3v, el cual es necesario para alimentar el sensor táctil capacitivo.

El Convertidor de Nivel Lógico Bidireccional (BD-LLC), es requerido para aislar la diferencia de voltajes de alimentación, ya que el microcontrolador se alimenta con 5 volts y el sensor con 3.3 volts.

2.1 Uso del Microcontrolador ATmega 328 mini.

Al tratarse de un dispositivo programable mediante computadora, la programación se efectúa a través de Arduino IDE, el cual es el software especial para programar dispositivos de este tipo, por lo que es posible el uso de la mayoría de librerías disponibles.

Para programar el dispositivo es indispensable contar con un convertidor USB serial, ya que Arduino Mini no cuenta con una interfaz USB propia. Por otra parte, es necesario realizar el quemado del bootloader del dispositivo cada vez que se requiera programarlo, de lo contrario resulta imposible programar el dispositivo (22).

La alimentación del microcontrolador requiere de una fuente de 5 volts, en este caso se utiliza una fuente con pila con esas características. La conexión del dispositivo a la alimentación y a los dispositivos principales, queda como lo muestra la Figura 2-2.

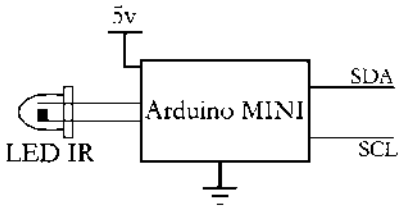


Figura 2-2 Conexión del Arduino MINI.

El microcontrolador incorpora el bus de comunicación I^2C, el cual es requerido en el diseño para establecer la comunicación con el sensor táctil capacitivo MPR121, ya que este sensor únicamente puede comunicarse por medio de este bus.

2.1.1 Programación usando el módulo PL2303.

Previo a conectar el Microntrolador con los demás dispositivos, es necesario programarlo, para ello se requiere el convertidor USB serial, con el cual es posible realizar la comunicación del microcontrolador y la computadora.

En este proyecto es usado el módulo PL2303, el cual convierte el puerto USB de la computadora en un puerto serial, con niveles de voltaje TTL (23), haciendo posible la comunicación entre la computadora y el microcontrolador.

La conexión de los dispositivos para llevar a cabo la programación, se muestra en la Figura 2-3.

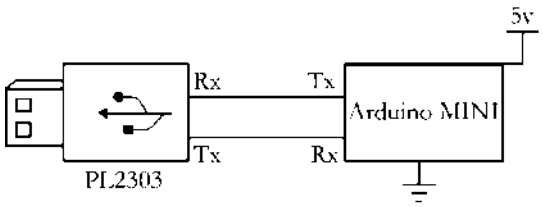


Figura 2-3 Conexión del módulo PL2303 y el Arduino MINI.

La Figura 2-3, muestra el diagrama de conexión adecuado para programar el Arduino Mini, únicamente son necesarias dos líneas de conexión, una para la transmisión de información (Tx) y otra para la recepción (Rx), estas deben estar conectadas de manera opuesta, es decir, mientras un dispositivo transmite información, el otro debe estar en modo de recepción, de lo contrario existe colisión en la información, provocando una programación incorrecta del microcontrolador.

2.1.2 Librería Wire.

Una de las utilidades que brinda el entorno Arduino IDE, es el uso de librerías, que facilitan la programación del microcontrolador ATmega 328. La librería Wire, incorpora funciones diseñadas para implementar el bus I^2C, mediante los pines de E/S en el microcontrolador (24).

Esta librería está encargada de establecer la comunicación entre la placa Arduino Mini, y el sensor MPR121. Para esto, únicamente se requieren dos líneas de comunicación, la línea de datos (SDA), que se encuentra en el pin analógico 4 del Microcontrolador, y la línea de reloj (SCL), ubicada en el pin analógico 5.

Las funciones necesarias para hacer uso de la librería son las siguientes:

- Wire.begin() y Wire.begin(dirección). Esta función inicializa la librería Wire y conecta el Arduino al bus. Si no se especifica la dirección, Arduino se conectará al bus como maestro, mientras que, si ésta se indica, lo hará como esclavo y asumiendo la dirección que se le ha proporcionado. En ambos casos, esta función no devuelve ningún valor.

- Wire.requestFrom(dirección, cantidad). Solicita bytes desde otros dispositivos.

El parámetro "dirección" es la dirección de 7 bits del dispositivo al que le queremos pedir los datos.

El parámetro "cantidad" es el número de bytes a pedir.

- Wire.beginTransmission(dirección). Comienza la transmisión con el dispositivo I^2C esclavo, en la dirección que se especifique.

El parámetro "dirección" es la dirección de 7 bits del dispositivo al que le queremos pedir los datos.

- Wire.endTransmission(). Finaliza la transmisión con el esclavo que fue comenzada por la función beginTransmission, y realmente lo que hace es transmitir los bytes que fueron obtenidos.

2.1.3 Librería IRremote.

Se trata de otra librería indispensable para el funcionamiento de Marcador IR, esta librería es quien permite la transmisión de códigos mediante infrarrojo, lo que permite que la herramienta se comporte como un mando a distancia.

Con anterioridad se mencionó que una de las funcionalidades más importantes del Marcador IR, es enviar códigos mediante infrarrojo, para posteriormente convertir esos códigos a eventos de ratón.

Esta librería puede ser operada de dos maneras; para transmitir códigos y para recibir códigos. En este caso, únicamente es usada para enviar códigos, los cuales cumplen con el protocoló de comunicación infrarrojo NEC (25).

Para realizar la transmisión únicamente es necesario incorporar la librería al bloque de programación en Arduino IDE, y de hardware solamente se requiere un LED infrarrojo, con su respectiva resistencia limitadora de 330 ohm, conectado en el pin 3 de la tarjeta de desarrollo Arduino Mini.

2.2 Convertidor de Nivel Lógico Bidireccional.

El Convertidor de Nivel Lógico Bidireccional (BD-LLC), representa un pequeño dispositivo electrónico encargado de asilar los dos niveles de voltaje diferentes en el bus de comunicación I^2C. Por un lado, encontramos el nivel alto de 5v con el que se alimenta el Arduino Mini, y por el otro el nivel bajo con el que es alimentado el sensor táctil capacitivo MPR121.

Los niveles lógicos que maneja este dispositivo son los siguientes:

- ✓ **Bajo:** 0v a 1v
- ✓ **Alto:** 3.5v a 5v

El dispositivo, está integrado por 4 circuitos de cambio de nivel, cada uno construido por un MOSFET de canal N y un par de resistencias pull-up, lo que permite un cambio de nivel bidireccional, esto se muestra en la Figura 2-4.

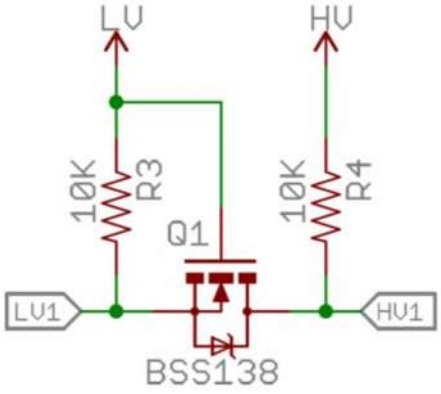


Figura 2-4 Circuito bidireccional de cambio de nivel.

Este circuito es capaz de cambiar una señal de bajo voltaje en una de alto voltaje, y viceversa, una de alto voltaje en otra de bajo voltaje. Por otro lado, una señal de 0v en cualquiera de los extremos sigue conservando su nivel de 0v. El dispositivo usado, cuenta con 4 circuitos idénticos al anterior.

La conexión de este dispositivo dentro de la herramienta, se muestra en la Figura 2-5.

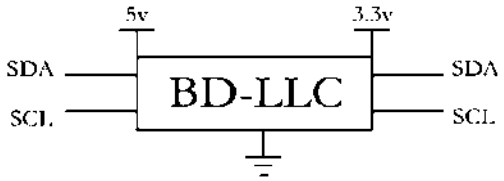


Figura 2-5 Conexión del Convertidor de Nivel Lógico bidireccional.

En la Figura 2-5, se observa la simplicidad de conexión del dispositivo, ya que solo se utilizan 2 de los 4 canales disponibles en el módulo, uno para la línea de datos (SDA) y el otro para la línea de sincronización (SCL). Así como, los dos diferentes voltajes de alimentación de los dispositivos a aislar.

Usando este módulo aseguramos la integridad de los dos dispositivos que están en comunicación, puesto que, en caso de algún corto circuito, este es soportado por las resistencias pull-up del módulo.

2.3 Sensor táctil capacitivo MPR121.

El sensor MPR121 es esencial en el diseño, ya que estará encargado de efectuar la interacción de la herramienta con el usuario.

Este sensor es capaz de detectar los cambios de capacitancia producidos sobre la superficie conectada a alguno de sus electrodos, esto lo hace por medio de umbrales de referencia, dichos umbrales están configurados en modo de activación, es decir, con solo tocarlos activan una función específica dentro de la herramienta, en este caso solo son usados cinco de sus trece electrodos disponibles, donde cada uno estará conectado a una superficie capacitiva.

El sensor admite un voltaje de alimentación de 3.3 volts, es el motivo por el cual se utiliza el convertidor de nivel lógico bidireccional.

Por otro lado, es indispensable usar una fuente regulada de CD a 3.3 volts, la cual, está encargada de alimentar el sensor, evitando dañarlo por sobrealimentación.

2.4 Fuente regulada CD de 3.3 volts.

Se trata de un pequeño regulador de corriente continua, con matricula Ams1117 (26), el cual viene montado en un mini tarjeta, la cual, incorpora dos filtros capacitivos. Este dispositivo permite un voltaje de alimentacion maximo de 18 volts, aunque para este proyecto será alimentado unicamente con 5v, que son los que proporciona la fuente con pila que se usa.

La conexión del sensor tactil capacitivo en conjunto con la fuente regulada se muestra en la Figura 2-6.

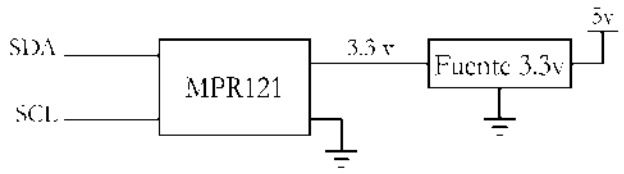


Figura 2-6 Conexión del sensor táctil capacitivo a la fuente regulada.

El marcador IR maneja dos valores de voltaje de alimentación diferentes, por un lado, el Arduino Mini es alimentado a 5v y por otro, el sensor táctil solo admite un voltaje de alimentación de 3.3 volts. El voltaje de 5 volts se obtiene directamente de la fuente con pila, pero para obtener el segundo voltaje de 3.3 volts es necesario disminuir el voltaje de la batería, por este motivo se requiere la fuente regulada.

Para evitar un mal funcionamiento en la conexión de los componentes de la herramienta, o en el peor de los casos dañar algún dispositivo, resulta muy conveniente el uso del convertidor de nivel lógico bidireccional, de esta forma se aíslan los dos diferentes niveles de voltaje.

2.5 Touchpad.

El Touchpad es el dispositivo encargado de la interacción física de la herramienta con el usuario. El diseño está pensado en obtener las funcionalidades de un mando a distancia infrarrojo, pero con respuesta a gestos de ratón.

El diseño está realizado sobre baquelita de cobre doble vista, con la finalidad de obtener mejor respuesta a la sensibilidad y a los cambios capacitivos

sobre la superficie. Así mismo facilita el manejo de la herramienta, evitando esfuerzo físico y mejorando la portabilidad del dispositivo.

El software utilizado para realizar este diseño es Eagle®, el cual está especializado en el diseño de circuitos electrónicos impresos, por lo que resulta sencillo el diseño del touchpad.

Figura 2-7 Vista superior del Touchpad.

Figura 2-8 Vista inferior del Touchpad.

Como se puede observar en la Figura 2-7, el Touchpad en su parte superior cuenta con tres botones de uso específico, los cuales asemejarán las características de los botones que integran un ratón de computadora, es decir, el clic izquierdo y el clic derecho, con un botón adicional para activar y desactivar el encendido de la herramienta. También cuenta con botón Slide, el cual adoptará la característica de la rueda Scroll integrada en un ratón común.

En la parte inferior de la baquelita de la Figura 2-8, únicamente se encuentran las pistas de conexión entre los botones mencionados y las cuales serán conectadas a los electrodos del sensor táctil.

La programación requerida para que la herramienta efectué correctamente las funciones mencionadas con anterioridad está escrita en

Arduino IDE, donde son incluidas las librerías Wire e Irremote, y tiene una estructura igual a como se muestra en el Pseudocódigo 1.

La librería IRremote brinda el comando sendNEC, con el cual se realiza el envió de comandos infrarrojos, lo único que requiere como argumento, es el comando a transmitir y la longitud de este, que por lo regular es de 32 bits.

Pseudocódigo 2-1 Marcador IR

Inicio

Si ⟶ Botón 1 tocado

Envía código NEC 1

Si ⟶ Botón 2 tocado

Envía código NEC 2

Si ⟶ Botón 3 tocado

Envía código NEC 3

Si ⟶ $\left(\begin{matrix}\text{Valor de capacitancia}\\ \text{boton 4}\end{matrix} = \text{crece}\right)$ Y $\left(\begin{matrix}\text{Valor de capacitancia}\\ \text{boton 5}\end{matrix} = \text{decrece}\right)$

Envía código NEC 4

Si ⟶ $\left(\begin{matrix}\text{Valor de capacitancia}\\ \text{boton 4}\end{matrix} = \text{decrece}\right)$ Y $\left(\begin{matrix}\text{Valor de capacitancia}\\ \text{boton 5}\end{matrix} = \text{crece}\right)$

Envía código NEC 5

Fin

2.6 Implementación del Marcador IR.

Una vez que se tienen diseñadas y probadas en protoboard cada etapa del marcador IR, es posible realizar la conexión de cada una de ellas y montarlas en baquelita, con la finalidad de evitar las fallas por cableado o ruptura de continuidad.

En la figura 2-9, se pueden observar cada uno de los bloques mencionados al principio del capítulo, con cada dispositivo electrónico correspondiente.

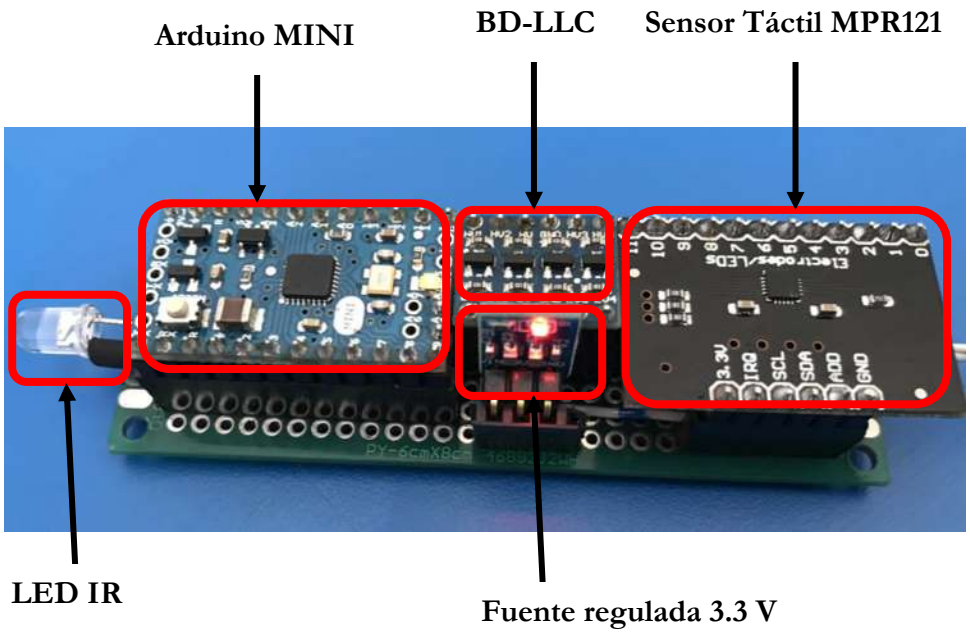


Figura 2-9 Implementación del Marcador IR.

Capítulo 3
Aplicación de escritorio

En el presente capítulo se aborda el diseño de la aplicación de escritorio, donde se incorporan algunos conceptos computacionales clave para el funcionamiento correcto del proyecto.

Se trata de una aplicación la cual debe ser ejecutada desde la computadora personal del usuario, en este caso, el usuario será el ponente. El esquema de funcionamiento de la aplicación, está pensado para funcionar en segundo plano, permitiendo al usuario continuar con sus tareas principales. Dicho de otra forma, el usuario ejecuta la aplicación de escritorio y se olvida de seguir interactuando con ella, puesto que la aplicación ejecuta sus tareas por su propia cuenta sin que el usuario tenga que intervenir. En la Figura 3-1, se muestra el diagrama de flujo que rige el funcionamiento de la aplicación de escritorio.

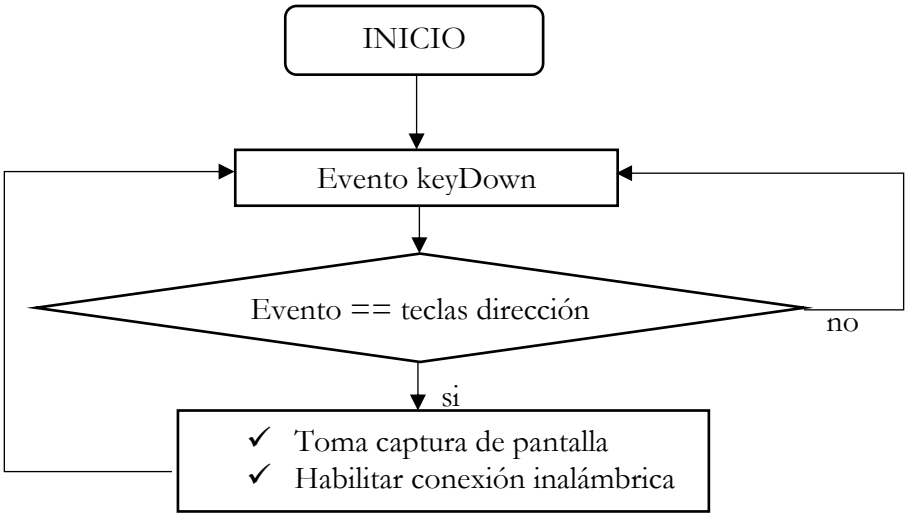


Figura 3-3-1 Diagrama de flujo de la aplicación de escritorio.

El inicio de la aplicación lo determinará el usuario, justo en el momento en el que decide ejecutar la aplicación, de ahí en adelante la aplicación se pone a la espera de un evento de teclado, el cual es provocado por mismo usuario cuando avanza y/o retrocede de diapositiva en la presentación, cuando esto sucede la aplicación efectúa las tareas siguientes en ese orden:

1. Realiza una captura a la pantalla de la computadora del usuario, y la sitúa en una carpeta compartida por Red con la Raspberry Pi asignándole un formato .jpg a la imagen.

2. Habilita una conexión vía Wifi mediante el uso de sockets con la Raspberry PI.

3. Envía una bandera a la aplicación que se ejecuta en Raspberry Pi, notificándole la existencia de una nueva captura en la carpeta compartida.

La aplicación de escritorio está diseñada para operar en el sistema operativo Windows®, ya que en este sistema operativo se encuentra el programa Power Point® en el cual se basa este proyecto. Este software es comúnmente utilizado por mayoría de los usuarios para llevar a cabo las presentaciones con diapositivas.

Las consideraciones de diseño de la aplicación, se basan en considerar únicamente las teclas que producen un cambio de diapositiva, hacia adelante o hacia atrás dentro de la presentación, y solo cuando el usuario haya iniciado la presentación de diapositivas.

El lenguaje de programación utilizado donde se diseña la aplicación, es C Sharp, el cual brinda la posibilidad de usar la programación dirigida por eventos y programación orientada a objetos.

3.1 Programación dirigida por eventos.

La aplicación de escritorio basa su funcionamiento en la programación dirigida por eventos, es decir, solamente toma acciones cuando el usuario provoca el evento deseado.

Este tipo de programación es totalmente opuesta a la programación secuencial o estructurada, donde el programador define cuál va a ser el flujo del programa, en la programación dirigida por eventos será el propio usuario o cualquier otro evento el que accione el programa, y a su vez el que dirija el flujo del mismo. Aunque en la programación secuencial puede haber intervención de un agente externo al programa, estas intervenciones ocurrirán cuando el programador lo haya determinado, y no en cualquier momento como en el caso de la programación dirigida por eventos (27).

3.2 Evento KeyDown.

El evento KeyDown forma parte de la programación orientada a eventos y es esencial en el funcionamiento de la aplicación de escritorio. Un evento KeyDown, es un evento externo, lanzado cuando una tecla es presionada hacia abajo, este evento es producido cuando el usuario se encuentra presionando teclas únicamente, cuando este deja de hacerlo el evento termina, y vuelve a entrar en funcionamiento únicamente cuando el usuario presiona las teclas nuevamente.

En la aplicación de escritorio, el evento KeyDown interviene únicamente cuando se presionan las teclas de dirección, directamente en el teclado de la computadora del usuario, ya que son estas las teclas que provocan un cambio de diapositiva, mientras estas teclas no se accionan, las demás teclas pueden seguir siendo presionadas pero la aplicación de escritorio no toma acciones.

3.3 Capturador de teclas (Keylogger).

Para que la aplicación de escritorio acceda a las entradas de teclado, es necesario un software capáz de reconocer cuando una tecla se ha presionado, para ello se requiere un software capturador de teclas.

Un capturador de teclas o Keylogger (28), es un software capaz de interceptar las pulsaciones del teclado de una computadora. En la actualidad un capturador de teclas está catalogado como un software malicioso por la facilidad que brinda de robar información personal, en este caso únicamente es requerido para identificar el momento en que fue presionada una tecla.

Este software se sitúa entre el teclado y el sistema operativo, con la finalidad de interceptar y registrar la información sin que el usuario lo note. Además, un keylogger es capáz de almacenar los datos de forma local en el ordenador si así se requiere, o en un caso más avanzado, permite acceso remoto al equipo teniendo acceso a la información desde otro equipo.

En la aplicación de escritorio el Keylogger es usado para monitorear los eventos KeyDown del teclado, y únicamente limitado para tomar acciones cuando las teclas de dirección o la tecla enter son presionadas. La figura 3.2, ilustra claramente cuáles son las teclas que monitorea el Keylogger.

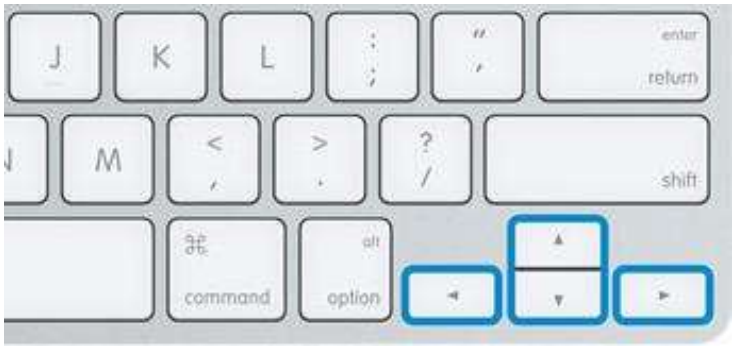


Figura 3-2 Teclas monitoreadas por e Keylogger.

La programación dirigida por eventos permite monitorear los eventos KeyDown provocados en las teclas seleccionadas, la implementación del keylogger y su funcionamiento se rigen bajo el Pseudocódigo 3-1.

Pseudocódigo 3-1 Keylogger

Inicio

Si evento→(tecla superior) O (tecla inferior) O (tecla siguiente) O (tecla anterior)

Hacer captura de pantalla

Habilita comunicación inalámbrica

Si evento→tecla escape

Cerrar aplicación

Fin

3.4 Capturador de pantalla (ScreenShot)

Una vez que la aplicación de escritorio detecto la presencia de un evento KeyDown en las teclas de dirección, entonces comienza a realizar sus tareas principales, entre ellas realizar la captura de pantalla en la computadora del usuario.

De este modo, la aplicación tiene acceso directamente a la diapositiva que está siendo presentada en ese momento. Con ello la aplicación guarda la captura, en una carpeta compartida por red con la Raspberry Pi, a partir de ahí la aplicación de escritorio sede el control de la imagen a la aplicación de la Raspberry, quien se encargará de poner esa captura en el plano de proyección.

La carpeta donde la aplicación de escritorio guarda la captura, se encuentra ubicada directamente en la Raspberry Pi, debido a que el sistema operativo Windows®, reconoce a la Raspberry como una simple carpeta contenedora de archivos. De este modo la aplicación de escritorio coloca la imagen de la captura de pantalla en la carpeta compartida, donde posteriormente la aplicación que se ejecuta en la Raspberry la tomará y efectuará sus tareas con ella.

El proceso de captura de pantalla dentro del entorno de programación C Sharp, es posible por la presencia del objeto *Graphics,* perteneciente a la clase *Bitmap* (29)*,* mismos que permiten realizar un mapa de bits del tamaño de la pantalla de la computadora y posteriormente crear el grafico en base al mapa de bits, en el Pseudocódigo 3-2, se muestra el proceso que se requiere seguir para implementar la función que efectúa la captura de pantalla.

Pseudocódigo 3-2 Capturador de pantalla

Inicio

Obtener resolución de la pantalla

Crear un mapa de bits del tamaño de la pantalla

Crear el grafico en base al mapa de bits

Guardar grafico en carpeta compartida

Fin

En el momento en que la captura de pantalla es tomada y guardada en la carpeta compartida, en otro hilo de la programación, la aplicación habilita la comunicación inalámbrica mediante Wifi, para notificar a la aplicación de la Raspberry Pi de la existencia de una nueva captura en la carpeta compartida.

3.5 Wi-Fi

El termino Wi-Fi, es el nombre de la marca registrada por la Wi-Fi Alliance para hacer referencia a la norma IEEE 802.11. Donde esta, solo es una red de área local inalámbrica o WLAN (30).

El alcance de estas redes inalámbricas varía dependiendo de la potencia con la que se transmite. Este alcance normalmente esta entre 100 y 200 metros, aunque inicialmente fue pensada para distancias cortas de 25 o 50 metros.

3.6 Sockets

Los sockets, son la interfaz más difundida para la comunicación de procesos. Socket, designa un concepto abstracto por el cual dos programas (situados en computadoras distintas) pueden intercambiarse cualquier secuencia de datos de manera transparente, sin conocer los detalles de cómo se transmiten esos datos, y generalmente de manera fiable y ordenada (31) .

Para que las aplicaciones puedan comunicarse entre sí, es necesario que una de ellas sea capaz de localizar a la otra, además, que ambas aplicaciones sean capaces de intercambiarse cualquier secuencia de datos.

Para ello son necesarios los tres recursos que originan el concepto de socket, y gracias a los cuales se definen:

- ✓ Un protocolo de comunicaciones, que permite el intercambio de datos.
- ✓ Una dirección del Protocolo de Red (dirección IP, si se utiliza el protocolo TCP/IP), que identifica una computadora.
- ✓ Un número de puerto, que identifica a un programa dentro de una computadora.

Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por una de las aplicaciones, siempre la que inicia la comunicación es la aplicación en modo cliente. Mientras que la otra aplicación espera a que el cliente inicie la comunicación, por este motivo se encuentra en modo servidor.

Desde el punto de vista de programación, un socket no es más que un archivo que se abre de una manera especial. Así un socket es un fichero existente en la máquina cliente y en la máquina servidor, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

La aplicación de escritorio, implementa la arquitectura tipo cliente, es decir, se encarga de establecer conexión e iniciar la comunicación con la aplicación de la Raspberry PI, donde esta última implementa la arquitectura de tipo servidor.

En C Sharp la creación de sockets se debe a la clase Socket presente en el espacio de nombres System.Net.Sockets. Esta clase tiene varios métodos y propiedades además de un constructor (31).

3.6.1 Creación

El primer paso es crear un objeto de esta clase usando el constructor (31).

3.6.2 Conexión

Una vez que hemos creado un socket, ahora necesitamos establecer una conexión con el servidor. Las dos aplicaciones, tienen primero que crear una conexión entre ellas. Las dos aplicaciones necesitarán identificar a la otra.

Para conectar con la computadora remota, en este caso la Raspberry Pi, necesitamos conocer la dirección IP y el puerto al cual conectar.

Si el servidor está funcionando y escuchando, la conexión tendrá éxito. Si en cambio el servidor no está operando, será lanzada una excepción. Asumiendo que la conexión está hecha, ya se puede mandar información al otro lado.

3.6.3 Sockets Asíncronos

Los sockets pueden ser síncronos y asíncronos, en este caso será utilizado el modelo de sockets asíncronos, la ventaja de usar este modelo radica en evitar el bloqueo de la aplicación, en el caso de los sockets síncronos la aplicación se bloquea hasta que se han recibido todos los datos (31).

Los sockets asíncronos solo requieren que se le pase como parámetro un buffer, que será donde se almacenen los datos recibidos, y una función que será llamada en cualquier momento que se reciban datos.

3.7 Cliente TCP/IP

La aplicación de escritorio utiliza el modelo cliente TCP/IP para establecer la comunicación con la aplicación de Raspberry Pi, de este modo, cuando la aplicación tiene una nueva captura de pantalla, envía una bandera que sirve de notificación al servidor que en este caso es la aplicación que se ejecuta

en la Raspberry Pi, indicándole de la existencia de la imagen de la captura de pantalla.

Cuando esto sucede la aplicación de escritorio nuevamente queda a la espera de que el usuario efectué un cambio de diapositiva para realizar una nueva captura a la diapositiva actual y notificárselo al servidor. De lo contrario, el cliente no vuelve a ser ejecutado hasta que el usuario provoque cambio en la diapositiva.

EL funcionamiento la aplicación de escritorio en modo cliente TCP/IP, se rige por el pseudocódigo 3-3.

Pseudocódigo 3-3 Cliente TCP/IP

```
Inicio
    Espera Evento KeyDown
    Si evento es provocado
        Realiza captura de pantalla
        Envía notificación al servidor
    Fin si
Fin
```

Capítulo 4
Ras Pizarrón.

En términos generales la aplicación que se ejecuta en la Raspberry Pi, es quien lleva a cabo la mayor parte del trabajo, ya que está encargada de establecer la interacción con cada uno de los elementos del proyecto, los cuales han sido mencionados en los capítulos anteriores.

La aplicación recibe el nombre de Ras Pizarrón, con la finalidad de no crear confusión con la Aplicación de Escritorio y a su vez, para identificarla como la aplicación que es ejecutada directamente en la Raspberry Pi. En la Figura 4-1, se muestra el esquema general de los tres procesos que lleva a cabo la aplicación y las tareas que efectúa cada uno.

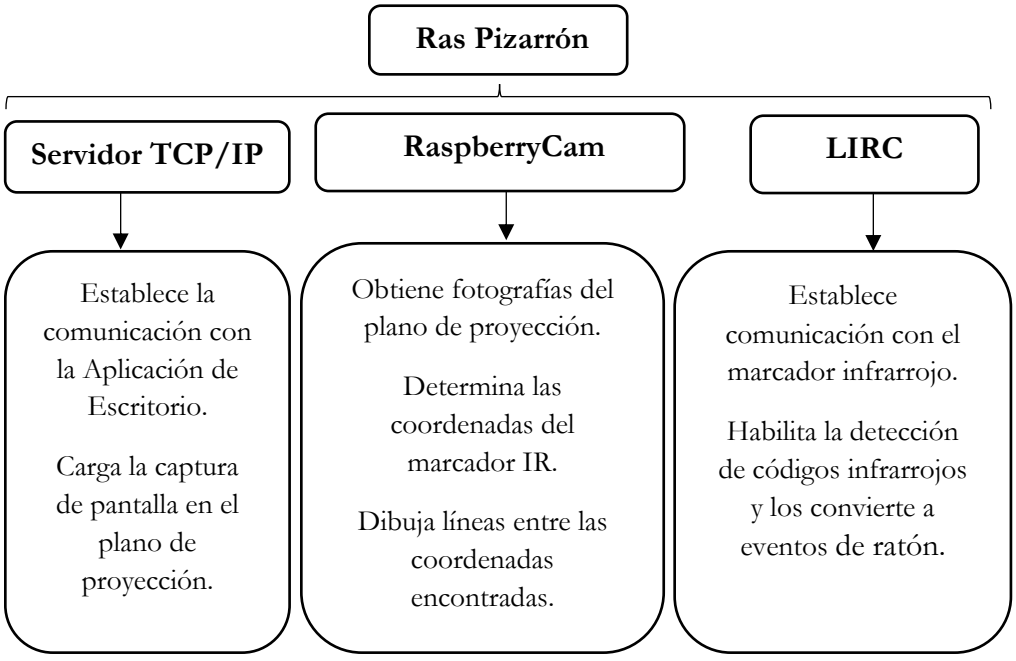


Figura 4-1 Diagrama general de Ras Pizarrón.

Principalmente la aplicación ejecuta tres tareas de forma paralela, la primera de ellas se encarga de habilitar la comunicación mediante Wi-fi con la aplicación de escritorio, otra se hace cargo de la captura de fotografías directamente al plano de proyección, con esto determinará la posición del marcador infrarrojo y por último habilita la comunicación vía sensor infrarrojo con el marcador IR y pone en marcha la detección de códigos, mismos que serán transformados en gestos de ratón.

Cada una de las tareas anteriores es posible debido al manejo de hilos en la programación de C Sharp, los hilos son procesos que se ejecutan en contextos diferentes y simultáneamente al proceso que crea el hilo (32), con ello se facilita correr los tres procesos que debe cubrir la aplicación.

Para que la aplicación lleve a cabo adecuadamente sus tareas, es necesario que la Raspberry Pi, cuente con algunos dispositivos periféricos adicionales, tales como:

- ✓ **Sensor de Cámara Pi Cam.**
- ✓ **Sensor infrarrojo 1838B en los puertos GPIO.**
- ✓ **Modulo Wifi USB TL-WN725N.**

El sensor de Cámara forma parte de los gadgets disponibles para Raspberry Pi, y es esencial para tomar las fotografías del plano de proyección y determinar la ubicación el marcador IR, lo cual permitirá calcular las coordenadas relativas a la imagen proyectada.

El sensor infrarrojo es un elemento comúnmente utilizado en dispositivos electrónicos manejados a control remoto, en está ocasión se encarga de detectar los comandos infrarrojos enviados por el marcador IR, los cuales serán procesados por la paquetería de LIRC y transformados en gestos de ratón.

El módulo WiFi es un dispositivo electrónico pequeño, con conexión vía USB, el cual permite a la Raspberry Pi establecer conexión a una red WiFi. Indispensable para que la comunicación con la aplicación de escritorio se lleve a cabo, de esta manera Ras Pizarrón obtiene acceso a la carpeta compartida donde la aplicación de escritorio guarda la imagen de la captura de pantalla.

Ras Pizarrón tiene acceso a la carpeta compartida haciendo uso del programa Samba, disponible para sistemas Linux el cual es una implementación del protocolo SMB (33). Este protocoló permite conectar dos computadoras en red, por lo que hace posible el intercambio de información entre los dos equipos, en este caso, los equipos conectados en red son la computadora personal del usuario, que es donde se ejecuta la aplicación de escritorio y la Raspberry Pi, donde se ejecuta Ras Pizarrón.

El diseño de Ras Pizarrón es una aplicación de tipo Interfaz de usuario (UI), la cual cuenta con algunos botones que permiten al usuario intuir su puesta en funcionamiento y decidir el momento preciso en que la aplicación se detenga.

4.1 Servidor TCP/IP

La comunicación de Ras Pizarrón y la Aplicación de Escritorio, es esencial para notificar inmediatamente la existencia de una nueva captura de pantalla hecha por la Aplicación de Escritorio, con ello, Ras Pizarrón se da cuenta que el ponente ha realizado un cambio de diapositiva en la presentación y que es momento de acceder a la carpeta compartida y colocar la nueva captura de pantalla en el plano de proyección, en este caso la captura de pantalla será la nueva diapositiva.

Ras Pizarrón, trabaja como servidor utilizando el protocolo TCP/IP, la cual queda a la espera de que el cliente (en este caso es la Aplicación de

Escritorio), inicie la comunicación entre ambas aplicaciones. El cliente envía notificaciones, solo cuando el usuario efectúa un cambio en la presentación. Una vez que el servidor de Ras Pizarrón recibe una notificación, este accede a la carpeta que se encuentra compartida mediante el sistema operativo donde se ejecuta la aplicación de escritorio, esto para extraer la captura de pantalla del ordenador y colocarla en el plano de proyección, donde se encuentra el usuario haciendo la presentación.

4.2 Interacción Cliente-Servidor

La interacción entre el cliente y el servidor, es de vital importancia para que ambas aplicaciones lleven a cabo una comunicación efectiva, es decir, la aplicación de Escritorio, la cual funciona bajo el modelo de cliente, es la encargada de lanzar la bandera de conexión con el servidor, el cual únicamente espera la bandera de inicio para comenzar a tomar acciones. De este modo ambas aplicaciones están en constante comunicación durante el proceso de ejecución. En la Figura 4-2, se muestra únicamente un barrido durante la ejecución de las aplicaciones, de la forma en cómo se lleva a cabo la interacción cliente-servidor.

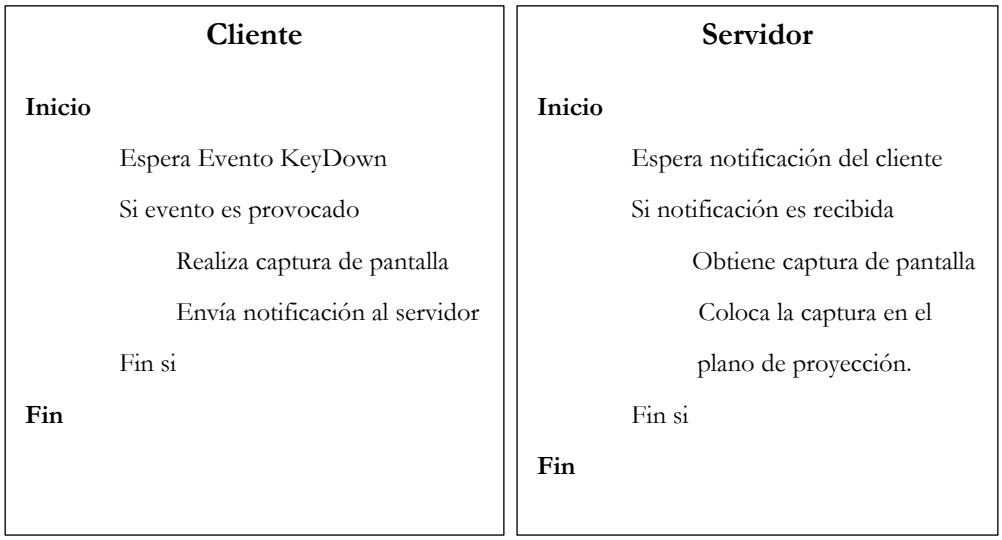


Figura 4-2 Interacción Cliente-Servidor.

Cabe mencionar, que el cliente entra en función únicamente cuando el usuario realiza un cambio de diapositiva, es decir, provoca el evento en las teclas seleccionadas, una vez hecha su tarea se pone a la espera nuevamente del evento. Y el servidor se acciona cuando el cliente envía la bandera de conexión, una vez que efectúa sus tareas, vuelve a esperar la bandera nuevamente.

4.3 Captura del plano de proyección

La importancia de obtener fotografías del plano de proyección, radica en conocer el estado actual del marcador IR, es decir, si el ponente acciono el marcador para dibujar sobre la diapositiva actual, puesta en el plano de proyección.

Para esto Ras Pizarrón, debe tener acceso al módulo de cámara Pi Cam, puesto en la Raspberry Pi. Por tal motivo se hace uso de la paquetería llamada RaspberryCam, la cual brinda acceso a la cámara desde Mono en .Net. Esta paquetería se encuentra disponible dentro de los paquetes de NuGet en Visual Studio (34). Para hacer uso de ella es necesario agregarla como referencia al diseño de Ras Pizarrón, y posteriormente mover el archivo .dll generado con el mismo nombre de la paquetería, a la carpeta de la Raspberry donde será ejecutada la aplicación.

Cuando Ras Pizarrón es ejecutada por primera ocasión después de encender la Raspberry, debe ser cargado el driver del módulo Pi Cam, para ello se hace uso de la instrucción siguiente:

Sudo modprobe bcm2835-v4l2

Una vez hecho lo anterior, el módulo está listo para comenzar a capturar fotografías, cuando Ras Pizarrón lo requiera.

Posteriormente, Ras Pizarrón requiere que el usuario le indique algunos parámetros para llevar a cabo su función correctamente.

4.3.1 Ajuste de esquinas

Ras Pizarrón antes de comenzar a tomar fotografías del plano de proyección, requiere que el usuario quien en este caso es el ponente, indique a la aplicación cuales son las cuatro esquinas de la diapositiva puesta en el plano de proyección, con esto la aplicación limita la búsqueda del marcador infrarrojo, únicamente al área dentro del cuadro formado por los puntos que marco el usuario.

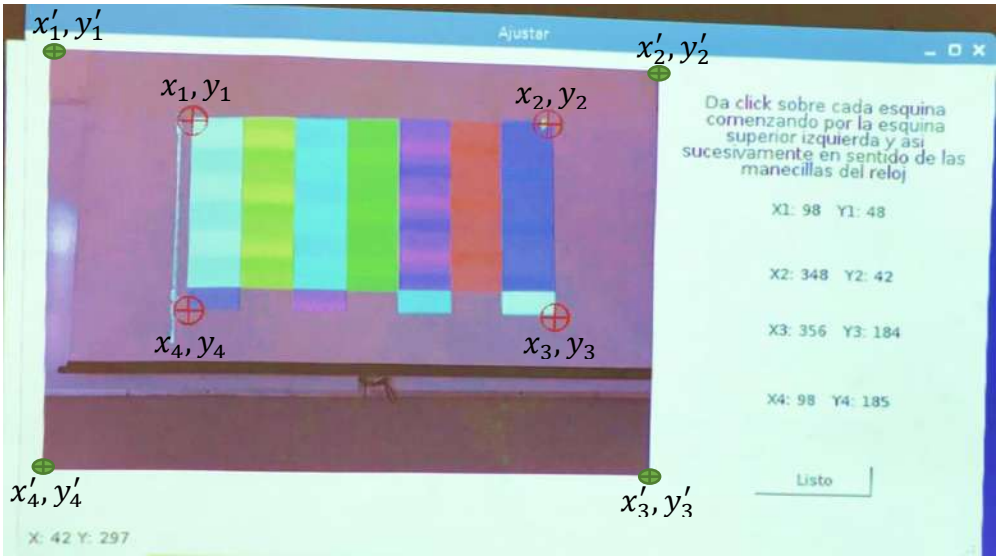


Figura 4-3 Ajuste homográfico.

En la Figura 4-3 se aprecia la forma de realizar el ajuste, se debe dar click sobre cada una de las esquinas hasta que aparezca el indicador rojo en forma de cruz, con ello Ras Pizarrón obtiene las coordenadas (x , y) que corresponden al plano de proyección. Posteriormente la aplicación efectúa la transformación afín

inversa de estas coordenadas, con la finalidad de ajustar el plano de proyección a las coordenadas ideales (x' , y') que aparecen con puntero color verde.

Para ello las coordenadas ideales se obtienen de la resolución del proyector, en este caso las coordenadas ideales son de 800*600 pixeles. Por lo tanto, la correspondencia de coordenadas para el ejemplo de la Figura 4-3, quedaría de la siguiente manera:

Coordenada del plano de proyección (x, y)	**Coordenada ideal** (x', y')
98,48	0,0
348,42	800,0
356,184	800,600
98,185	0,600

Tabla 4-1 Correspondencia de coordenadas.

Lo anterior es indispensable para determinar la correspondencia entre el plano de proyección y el proyector de video o la pantalla dependiendo el caso.

Con lo anterior, Ras Pizarrón aplica la transformación afín inversa, usando una matriz de homografía (17), de esta manera no importa si el sensor de cámara se encuentra desalineado respecto al plano de proyección, con indicarle cuales son las esquinas, la aplicación ajusta las coordenadas de cada esquina a las coordenadas ideales.

El ajuste de esquinas evita que la aplicación subraye fuera del plano de proyección, permitiendo la eficiencia de la aplicación.

4.3.2 Iniciar Ras Pizarrón

Cuando el usuario terminó de indicarle a Ras Pizarrón, las esquinas del plano de proyección, entonces es posible iniciar la captura de fotografías de este, lo cual, es necesario para determinar las coordenadas del marcador infrarrojo el cual se encuentra siendo operado por el ponente.

Una vez iniciada la aplicación, esta lleva a cabo una secuencia de operación muy estricta, es decir, realiza un barrido a la vez y no se sigue ejecutando hasta que termino completamente con el barrido anterior, esto es posible mediante la programación asíncrona (35).

Primeramente, Ras Pizarrón obtiene una fotografía del plano de proyección, posteriormente esta fotografía es transformada a una imagen en escala de grises, donde resulta más fácil encontrar el Marcador IR, puesto que, en esta escala, el infrarrojo tiene un umbral muy alto que hace que se vea de color blanco muy brillante.

La secuencia de operación, se puede entender más claramente con el siguiente seudocódigo:

Pseudocódigo 4-1 Capturador del plano de proyección

```
Nombre: Capturador de plano de proyección
Inicio
        Ajustar el plano de proyección mediante homografía
        Obtener fotografía del plano de proyección
        Aplicar escala de grises a la fotografía
        Buscar el pixel más brillante
Fin
```

4.3.3 Búsqueda del pixel más brillante

La ubicación del pixel más brillante, es quien determinará las coordenadas del Marcador IR, dentro de la fotografía del plano de proyección.

Para determinar la búsqueda del pixel más brillante dentro de la imagen, ésta es tratada como una matriz de dos dimensiones, donde, cada elemento de la matriz corresponde a un pixel con su respectiva combinación de colores RGB.

Para ubicarlo se realiza el barrido completo de la fotografía, haciendo la comparación del valor de umbral de cada uno de los pixeles, con el umbral de referencia, el cual tiene un valor mayor o igual a 240.

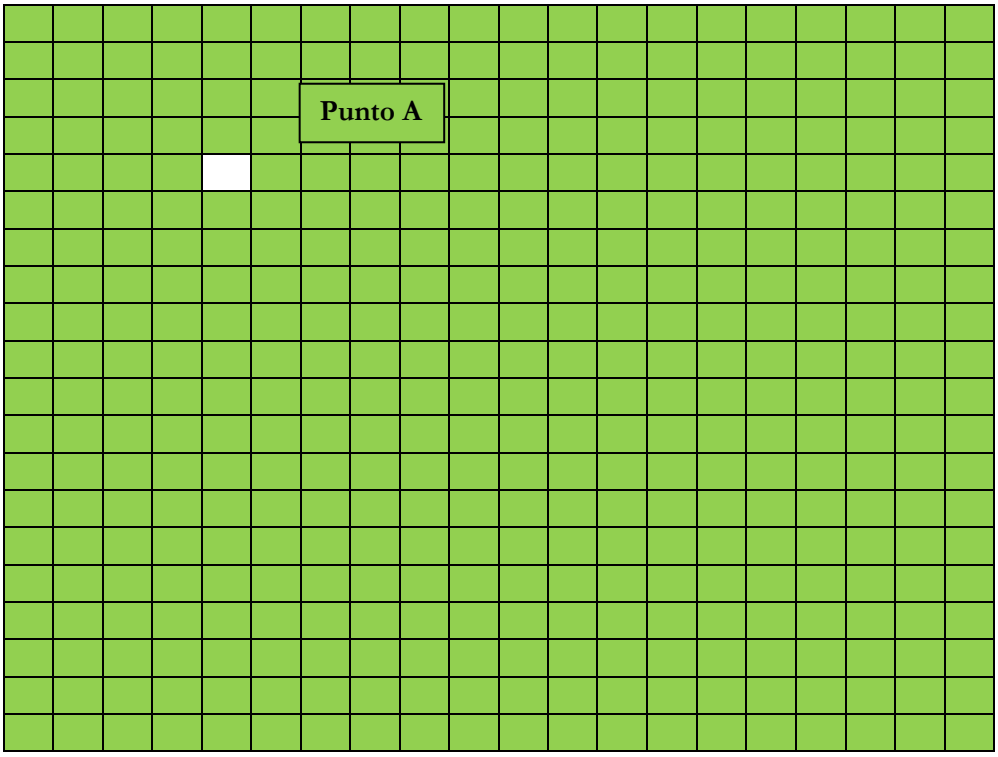


Figura 4-4 Búsqueda del pixel más brillante.

La Figura 4-3, presenta el caso ideal de la forma como son tratadas las fotografías del plano de proyección, cada recuadro representa un pixel de la

imagen, donde el recuadro en color blanco representa el pixel con el umbral requerido, y el cual, es producido por la activación del marcador IR.

Si Ras Pizarrón, encuentra un pixel brillante, entonces determina la homografía de ese punto, para obtener las coordenadas correspondientes al plano de proyección, una vez hecho esto, dibuja un punto en esa coordenada.

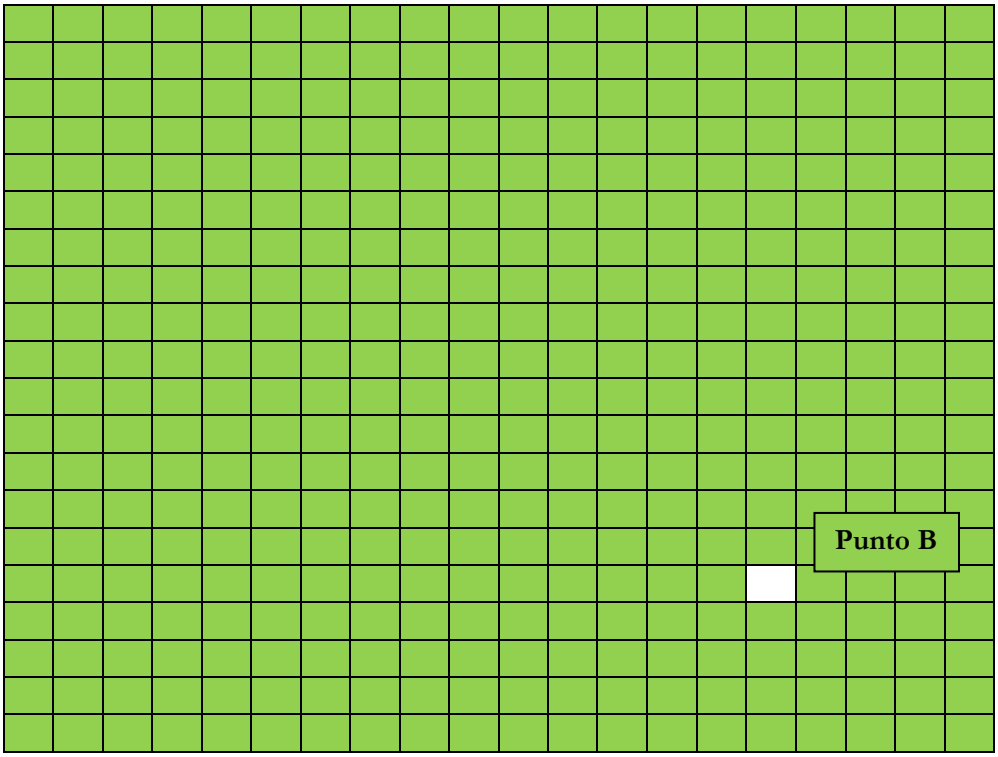


Figura 4-5 Nuevo pixel brillante.

Si después de varias fotografías, Ras Pizarrón determina que existe otro pixel donde el umbral es del valor requerido, entonces nuevamente calcula la homografía de ese punto y determina las coordenadas del nuevo punto, para posteriormente dibujar una línea entre el punto A y el nuevo punto B. Tal como se muestra en la Figura 4-5.

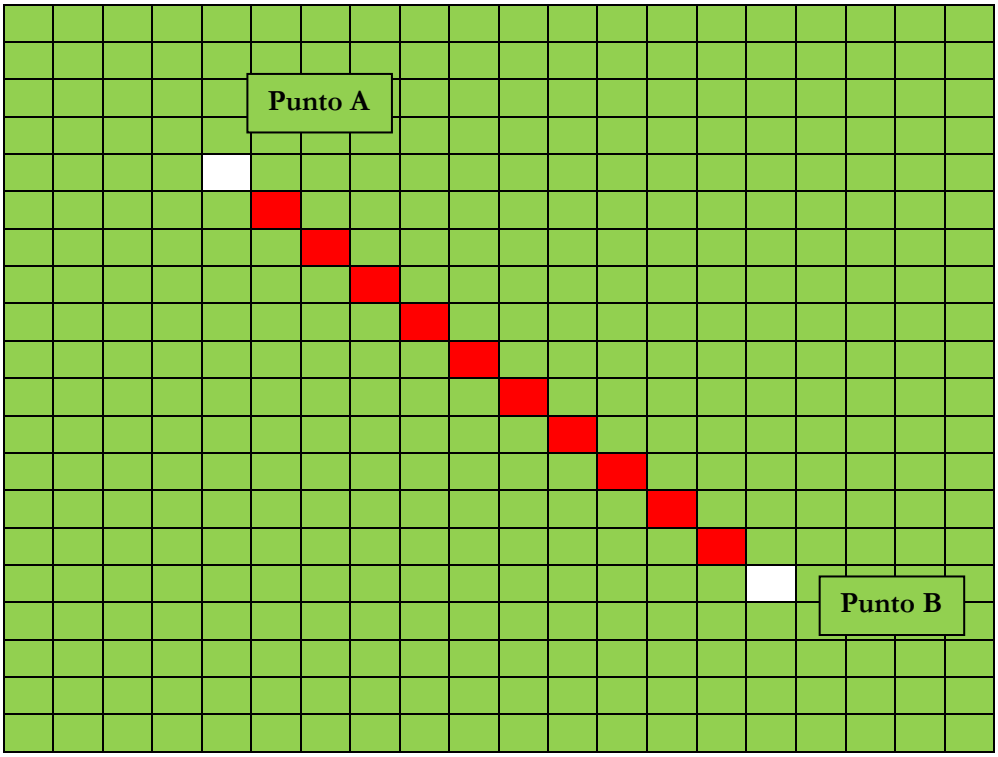


Figura 4-6 Trazado de línea entre A y B.

De esta manera Ras Pizarrón, va efectuando el mismo proceso de búsqueda a cada una de las fotografías tomadas, dándole el tiempo necesario a cada imagen, es decir, no inicia a tomar una nueva fotografía hasta que termina de analizar la anterior, con ello verifica cuidadosamente la activación o no del Marcador IR. Y a su vez, determina mediante homografía, la coordenada que muestra la posición del mismo.

4.4 Detección de códigos IR

Como se mencionó dentro de los objetivos, el marcador IR debe cumplir con los gestos funcionales de un ratón convencional de computadora, es decir, clic izquierdo, clic derecho y las funciones de scroll up/down. Para ello se utiliza la paquetería LIRC.

Con esta paquetería, es posible la detección de códigos de cualquier tipo de mando a distancia infrarrojo, siempre y cuando cumplan con alguno de los protocolos de comunicación infrarroja, en este caso, es utilizada para detectar los códigos infrarrojos enviados por el Marcador IR, el cual está configurado para enviar diferentes códigos para cada uno de los botones que lo integran.

Para esto, resulta necesario hablar de los códigos con los que están configurado cada uno de los botones del Marcador IR, la Figura muestra los códigos IR que son enviados al presionar el botón correspondiente:

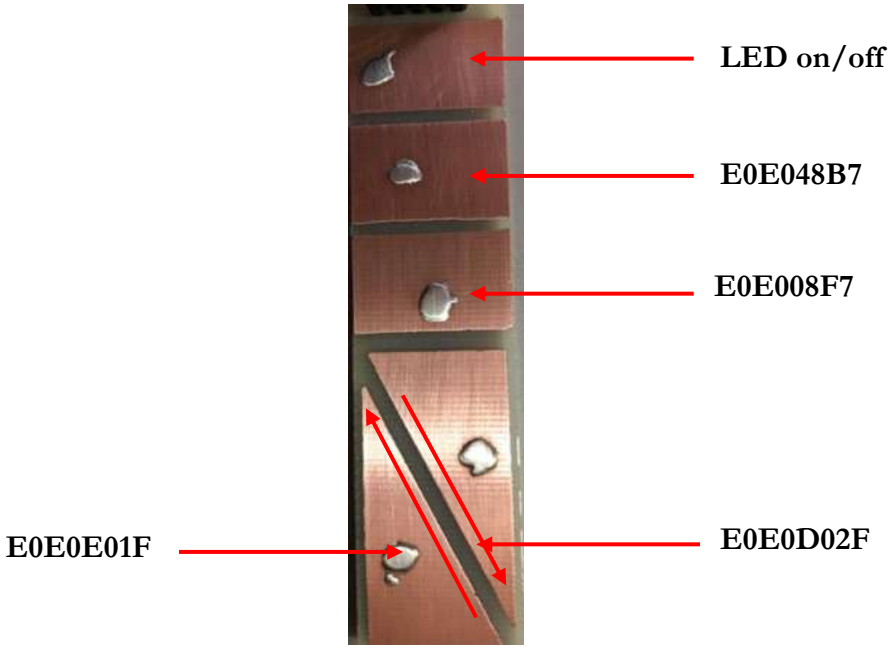


Figura 4-7 Asignación de códigos NEC a los botones del Marcador IR.

En LIRC, es necesario grabar los códigos anteriores del Marcador IR, para esto lirc incluye el programa Irrecord, que permite reconocer las señales infrarrojas y asignarles un nombre de botón a cada señal. El programa se encarga de generar el archivo de configuración lircd.conf, en el cual quedan asignados los nombres de cada botón. La Figura 4-7, se muestra el archivo generado después de grabar las señales infrarrojas.

```
  GNU nano 2.7.4                File: lircd.conf

# Please take the time to finish this file as described in
# https://sourceforge.net/p/lirc-remotes/wiki/Checklist/
# and make it available to others by sending it to
# <lirc@bartelmus.de>
#
# This config file was automatically generated
# using lirc-0.9.4c(default) on Tue Feb 13 23:16:51 2018
# Command line used: -d /dev/lirc0 /home/pi/lircd.conf
# Kernel version (uname -r): 4.9.59+
#
# Remote name (as of config file): MarcadorIR
# Brand of remote device, the thing you hold in your hand:
# Remote device model nr:
# Remote device info url:
# Does remote device has a bundled capture device e. g., a
#     usb dongle? :
# For bundled USB devices: usb vendor id, product id
#     and device string (use dmesg or lsusb):
# Type of device controlled
#     (TV, VCR, Audio, DVD, Satellite, Cable, HTPC, ...) :
# Device(s) controlled by this remote:

begin remote

  name  MarcadorIR
  bits           16
  flags SPACE_ENC|CONST_LENGTH
  eps            30
  aeps          100

  header       4533  4509
  one           583  1683
  zero          583   560
  ptrail        586
  pre_data_bits   16
  pre_data       0xE0E0
  gap          108811
  toggle_bit_mask 0x0
  frequency    38000

      begin codes
          BTN_RIGHT                0xE01F
          BTN_LEFT                 0xD02F
          KEY_SCROLLUP             0x48B7
          KEY_SCROLLDOWN           0x08F7
      end codes

end remote
```

Figura 4-8 Configuración del demonio lircd.conf.

4.5 Asignación de los códigos a gestos de ratón

Para LIRC, los códigos anteriores son reconocidos como un control remoto común, posteriormente cado uno de estos, es asignado al sistema operativo por LIRC como un evento de ratón, es decir, que cada código responde a un evento de ratón. En la figura siguiente se observa la asignación de cada código a un gesto de ratón.

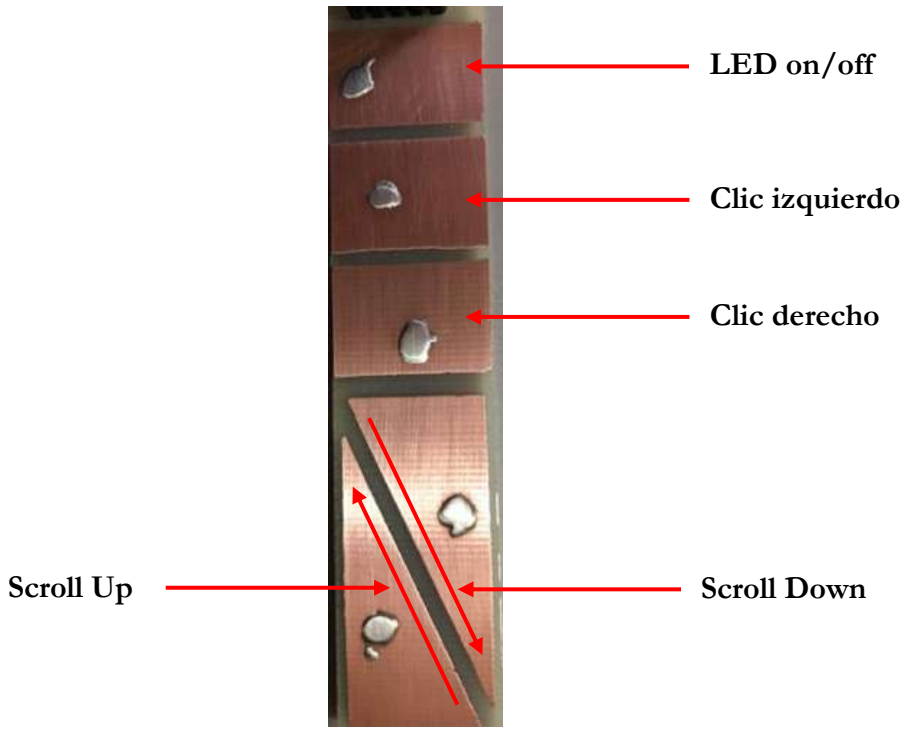


Figura 4-9 Asignación de eventos de ratón para cada código NEC.

En la Figura 4-8, se muestran los eventos de ratón a los cuales responde cada botón en el Marcador IR.

En LIRC, es necesario configurar el demonio Lircmd, que incluye el software necesario para hacer esta traducción de simples señales infrarrojas a eventos de ratón. Para ello se tiene que configurar otro archivo, el cual recibe el nombre de lircmd.conf.

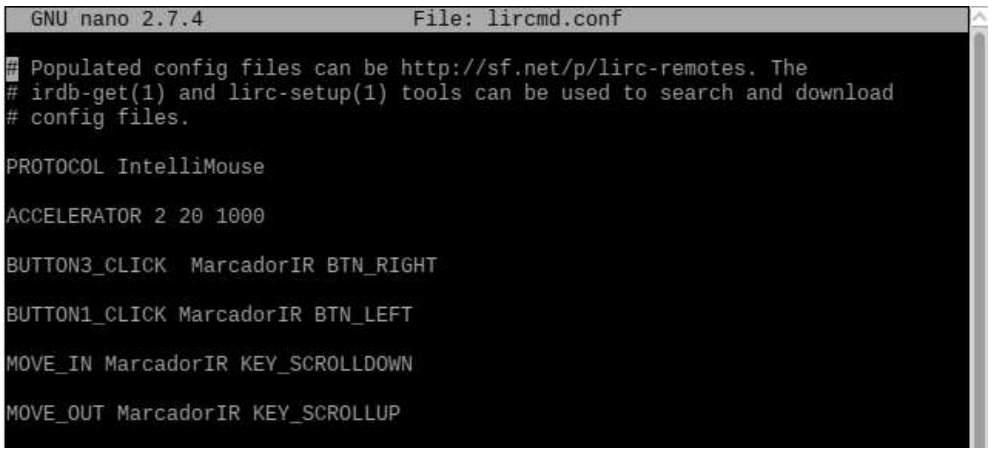


Figura 4-10 Configuración del demonio Lircmd.conf.

Una vez configurados cada uno de los códigos infrarrojos como gestos de ratón es posible utilizar el Marcador IR de igual manera que un ratón convencional, ya que responde al click izquierdo, click derecho y simula la rueda scroll.

Capítulo 5
Pruebas y Resultados.

5.1 Pruebas

La prueba de visualización surge con la necesidad de observar la intensidad con que emite el marcador IR y comprobar la visibilidad del sensor de cámara Pi Cam. Así mismo demostrar que la radiación infrarroja sobresalga en cualquiera de los colores puestos en pantalla, para ello se utiliza la típica pantalla con barras de colores.

Figura 5-1 Prueba de visualización en un monitor.

La Figura 5-2 deja ver la prueba de visualización directamente en un monitor, donde se aprecia claramente la radiación emitida por el LED infrarrojo del marcador IR, lo suficientemente brillante y sobresaliente de los demás colores facilitando el trabajo de Ras Pizarrón para ubicar el pixel más brillante en la fotografía capturada.

5.2 Resultados

Con la finalidad de demostrar el funcionamiento de la herramienta, se realizan las pruebas que determinan las condiciones aptas donde puede ser utilizada la herramienta.

Condiciones del espacio donde se probó la herramienta:

- ✓ Salón de clases con poca iluminación.
- ✓ Cortinas para evitar el paso de la luz solar.
- ✓ Proyector de la marca ViewSonic con resolución de 800x600 pixeles.
- ✓ La posición del proyector es en el techo del salón a una altura aproximada de 2.5 metros.
- ✓ Altura de 1.5 metros entre el piso y la posición de la cámara Pi Cam y la Raspberry Pi.
- ✓ Distancia horizontal de 2 metros entre la cámara de la Raspberry Pi y el plano de proyección.
- ✓ La cámara debe visualizar el plano de proyección en su totalidad.
- ✓ Red Wi-Fi con muy buena recepción, con la finalidad de establecer la comunicación Cliente-Servidor.
- ✓ Alimentación a 120 Volts de AC.

Una vez que se cumple con las condiciones anteriores, la herramienta funciona correctamente de acuerdo a lo planeado. Lo que sigue es poner en marcha la herramienta únicamente cuidando el orden de ejecución de cada uno de los elementos que la integran.

Primero es necesario ejecutar Ras Pizarrón en la Raspberry Pi, en ese momento aparecerá una aplicación de tipo interfaz de usuario como se muestra la Figura 5-2.

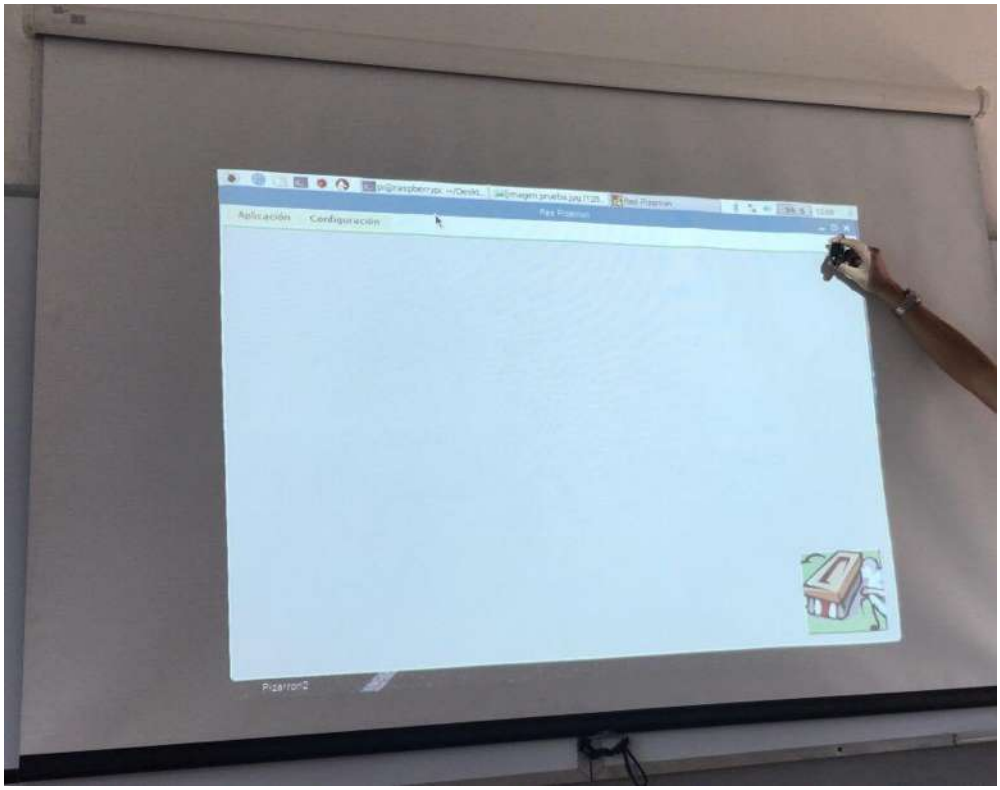


Figura 5-2 Ras Pizarrón.

Una vez puesta en marcha la aplicación, se procede a realizar el ajuste de esquinas correspondiente, limitando el espacio de subrayado únicamente al área entre las coordenadas seleccionadas. Esta opción aparece en la pestaña configuración, la Figura 5-3 muestra la ventana de ajuste.

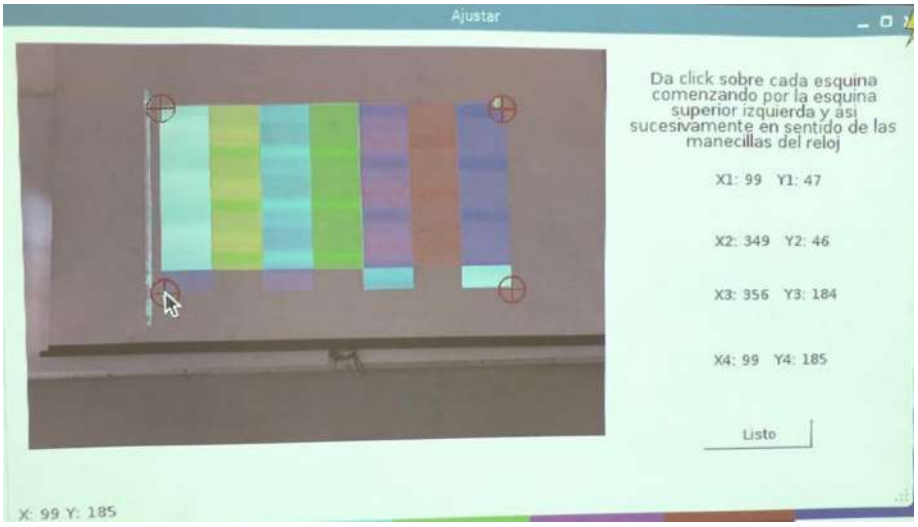


Figura 5-3 Ajuste de esquinas.

La ventana ajustar muestra las indicaciones necesarias para llevar a cabo el ajuste correctamente. Se debe comenzar por seleccionar la esquina superior izquierda y posteriormente continuar seleccionando en sentido horario, es decir, la esquina siguiente en seleccionar sería la esquina superior derecha y así sucesivamente hasta llegar a la esquina inferior izquierda.

En cuanto el ajuste se encuentra listo, el usuario puede decidir el instante en que la aplicación inicia la búsqueda del marcador Ir, ya que en la pestaña de "aplicación" se encuentra la opción de iniciar y la opción detener.

Por otro lado, en la computadora personal del usuario ya puede ser ejecutada la aplicación de escritorio, lo único que se debe establecer en dicha aplicación, es la dirección IP de la Raspberry Pi donde se ejecuta Ras Pizarrón. Una vez hecho esto, ya es posible observar la conectividad inalámbrica entre ambas aplicaciones, es decir, cualquier acción que involucre presionar las teclas de dirección en el teclado de la computadora efectuara el cambio de imagen en Ras Pizarrón y por consiguiente en el plano de proyección. Las Figuras 5-4 y 5-5, demuestran la conectividad inalámbrica entre las aplicaciones.

Figura 5-4 Comprobación de conectividad inalámbrica entre las aplicaciones.

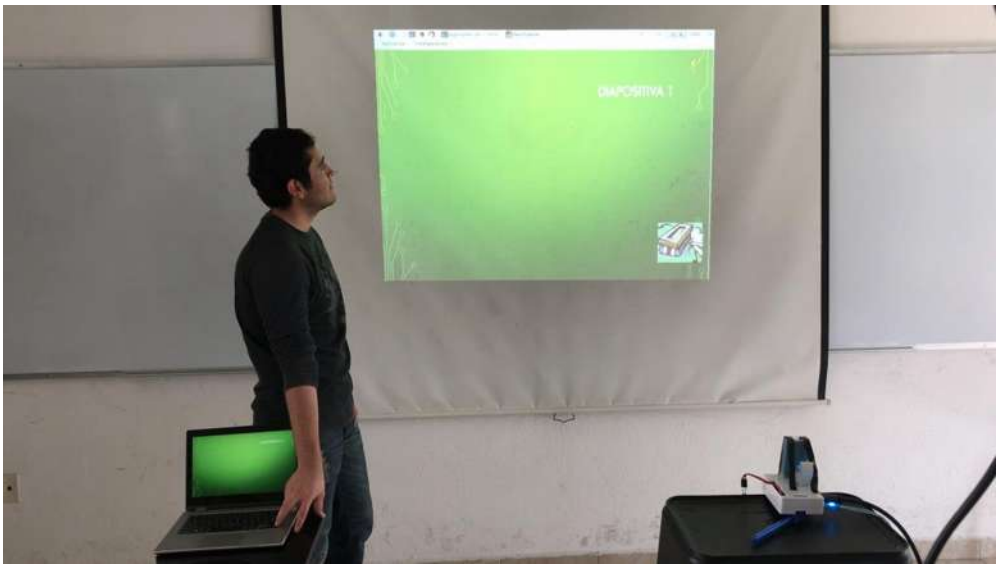


Figura 5-5 Cambio de diapositiva.

Cuando ambas aplicaciones ya se encuentran en ejecución y el usuario inicia la búsqueda, entonces el marcador Ir puede ser accionado en cualquier parte del plano de proyección dentro de las coordenadas indicadas, con ello el usuario

tiene la posibilidad de comenzar a subrayar o a dibujar sobre la diapositiva. La Figura 5-6 muestra la herramienta en pleno funcionamiento.

Figura 5-6 Trazado de un polígono usando la herramienta.

Figura 5-7 Subrayado usando la herramienta.

En la Figura 5-7 se observa el funcionamiento de la herramienta, donde se aprecia un tiempo de retraso entre la posición del marcador Ir y el trazado de la línea, esta diferencia de tiempo corresponde al tiempo que se lleva la aplicación Ras Pizarrón para efectuar el procesamiento de detección de las coordenadas del marcador.

Este retraso de tiempo puede ser corregido, si se buscan algoritmos de detección más eficientes o en su defecto llevar a cabo toda la programación directamente en la GPU (Unidad Grafica de Procesamiento por sus siglas en ingles) de la Raspberry Pi.

Con lo anterior la velocidad de procesamiento mejoraría, lo que haría que la herramienta operara de una manera más eficiente.

Capítulo 6
Conclusiones y trabajos futuros.

6.1 Conclusiones

El proyecto desarrollado en esta tesis, cumple con los objetivos planteados al principio, cada uno de los elementos de la herramienta cubren las necesidades requeridas por la herramienta, aunque a lo largo del camino algunos dispositivitos fueron sustituidos por otros, e incluso la manera de desarrollar los algoritmos también cambio, pero el objetivo siempre permaneció intacto.

El Marcador IR cumple con las condiciones propuestas, una vez puesto en marcha se configuró con los comandos necesarios para encender y controlar el volumen en una televisión, comprobando el funcionamiento del dispositivo como mando a distancia en cualquier otro aparato controlado por infrarrojo.

La aplicación de Escritorio también cumple con sus requerimientos, efectuando la intercepción de las teclas de dirección y llevando a cabo la tarea de captura de pantalla y posteriormente la comunicación inalámbrica con Ras Pizarrón.

Por otro lado, la herramienta maneja diferentes entornos, es decir, no se limita únicamente a uno, sino que aborda conceptos electrónicos y computacionales con la suficiente importancia para la formación académica.

Con la implementación de este proyecto de tesis se adquieren nuevos conocimientos sobre cómo manejar dispositivos electrónicos en la solución problemas tecnológicos, así mismo, el manejo de un lenguaje de programación cuya base de usuarios ha crecido bastante en los últimos años. De este modo, el

proyecto brinda la oportunidad de continuar investigando sobre el manejo de Raspberry Pi y sus utilidades en el área de visión computacional.

6.2 Trabajos futuros

Es interés del autor continuar con el desarrollo de la herramienta, debido a que con este proyecto de tesis únicamente se demuestra el funcionamiento del prototipo, faltaría mejorar la velocidad de procesamiento y en general la velocidad operación en la herramienta.

Controlar el manejo de todos los eventos que provoquen un cambio de diapositiva, es decir, analizar cada evento dentro del programa Power Point®, con el fin de anexarlos a las funcionalidades de la herramienta.

Implementar algoritmos de detección del pixel más brillante ejecutados directamente en la GPU (Unidad Grafica de Procesos) del Raspberry Pi, con esto se espera aumentar la velocidad de procesamiento de las imágenes en Ras Pizarrón.

Utilizar la transferencia de imágenes vía stream de datos, con esto, se espera agilizar el proceso de comunicación ente las dos aplicaciones, ya que, el tiempo de transferencia de las capturas de pantalla sería más ágil, es decir, se estaría evitando el uso de un servicio extra (samba).

Por otro lado, también se pretende incluir el conjunto de trasformaciones ortogonales a la herramienta, con la finalidad de determinar con mayor precisión las coordenadas del marcador Ir.

Anexo A. Código fuente marcador Ir

```
#include <Wire.h>
#include <IRremote.h>
#include <IRremoteInt.h>
#include <Adafruit_MPR121.h>

Adafruit_MPR121 cap = Adafruit_MPR121();
uint16_t lasttouched = 0;
uint16_t currtouched = 0;
Irsend;
const int LED = 3;


void setup() {
    if (!cap.begin(0x5A)) {                    // 0x5A es la dirección del sensor MPR121
          while (1);
    }
     pinMode(LED,OUTPUT);                      //Establece el pin del LED como salida
}



void loop() {
   currtouched = cap.touched();                // Obtiene el electrodo que fue tocado
   for (uint8_t i=3; i<9; i++) {               // Monitorea el estado de los electrodos
         if ((currtouched & _BV(i))   &&   !(lasttouched & _BV(i)) ) {   //Tocado
            if(i=3){
                digitalWrite (LED,HIGH);                    //LED On
                delay(1000);
             }
            if(i=4){
                irsend.sendNEC(0xE0E0D02F,32);              //Envía código Ir
                delay(100);
            }
            if(i=5){
                irsend.sendNEC(0xE0E0E01F,32);               //Envía código Ir
                delay(100);
            }
           while ((i=6) && (i=7)){
           if(cap.filteredData(7)>20    &&    cap.filteredData(8)>20){
                irsend.sendNEC(0xE0E048B7,32);              //Envía código Ir
                 delay (40);
            }
            if (cap.filteredData(7)<20 && cap.filteredData(8)<20){
                irsend.sendNEC(0xE0E008F7,32);              //Envía código Ir
                delay(40);
           }
           return;
        }
  }
```

```
      if (!(currtouched & _BV(i))   &&   (lasttouched & _BV(i)) ) {   // Si ha sido soltado
          digitalWrite (LED,LOW);                                     //LED off
       }
      }
      lasttouched = currtouched;                                  // resetea el estado del sensor
      return;
}
```

Anexo B. Código fuente para la aplicación de escritorio

```
using System;
using System.Diagnostics;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading;
using System.Windows.Forms;

public class InterceptKeys
{
   private const int WH_KEYBOARD_LL = 13;
   private const int WM_KEYDOWN = 0x0100;
   private static LowLevelKeyboardProc _proc = HookCallback;
   private static IntPtr _hookID = IntPtr.Zero;

   public static void Main(string[] args)
   {
      _hookID = SetHook(_proc);
      Application.Run();
      UnhookWindowsHookEx(_hookID);
   }

   //Función que configura la aplicación para ejecutarse en segundo plano
   private static IntPtr SetHook(LowLevelKeyboardProc proc)
   {
      using (Process curProcess = Process.GetCurrentProcess())
      using (ProcessModule curModule = curProcess.MainModule)
      {
         return SetWindowsHookEx(WH_KEYBOARD_LL, proc,
            GetModuleHandle(curModule.ModuleName), 0);
      }
   }

   private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);

   //Función que monitorea los eventos KeyDown en el teclado
   public static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
   {
      if (nCode >= 0 && wParam == (IntPtr)WM_KEYDOWN)
      {
         int vkCode = Marshal.ReadInt32(lParam);
         if (Keys.Down == (Keys)vkCode || Keys.Up == (Keys)vkCode || Keys.Right ==
            (Keys)vkCode || Keys.Left == (Keys)vkCode || Keys.Enter == (Keys)vkCode)
         {
```

```
            ThreadStart delegado = new ThreadStart(Screenshot); //Creamos la instancia a un nuevo hilo
            Thread hilo = new Thread(delegado);                 //Ejecutará la función de captura pantalla
            hilo.Start();                                       //Iniciamos el hilo
            hilo.Join();                                        //Esperamos a que termine

            ThreadStart delegado1 = new ThreadStart(StartClient);//Creamos la instancia a otro hilo
            Thread hilo1 = new Thread(delegado1);               //Ejecutará la función del cliente TCP/IP
            hilo1.Start();                                      //Inicia el hilo
            hilo1.Join();                                       //Espera a que termine
        }

        if (Keys.Escape == (Keys)vkCode)
        {
            Application.Exit();
        }
    }

    return CallNextHookEx(_hookID, nCode, wParam, lParam);
}

//Función encargada de efectuar la captura de pantalla en la computadora
private static void Screenshot()
{
    try
    {
        Thread.Sleep(250);
        Bitmap bmpScreenShot=new Bitmap(Screen.PrimaryScreen.Bounds.Width,
        Screen.PrimaryScreen.Bounds.Height);                //Nuevo mapa de bits

        Graphics ScreenShot = Graphics.FromImage(bmpScreenShot);
        ScreenShot.CopyFromScreen(Screen.PrimaryScreen.Bounds.X,  Screen.PrimaryScreen.Bounds.Y,
        0, 0, Screen.PrimaryScreen.Bounds.Size);            //Nuevo grafico con las dimensiones de la pantalla

        bmpScreenShot.Save(@"\\RASPBERRYPI\ScreenShot\home\pi\Desktop\CapretaC
        ompartidaPI\ScreenShot.png");                       //Ubicación de red donde se guarda la imagen de la captura

    }
    catch (Exception objError)
    {
        MessageBox.Show(objError.ToString(), "Error", MessageBoxButtons.OK,   MessageBoxIcon.Error);
    }
}

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
 private static extern IntPtr SetWindowsHookEx(int idHook, LowLevelKeyboardProc lpfn,
 IntPtr hMod, uint dwThreadId);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
[return: MarshalAs(UnmanagedType.Bool)]
private static extern bool UnhookWindowsHookEx(IntPtr hhk);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
```

```
    private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr
lParam);

    [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
    private static extern IntPtr GetModuleHandle(string lpModuleName);

    [DllImport("kernel32.dll")]
    private static extern IntPtr GetConsoleWindow();

    [DllImport("user32.dll")]
    private static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);

    private const int SW_HIDE = 0;

 //BLOQUE DE CODIGO PARA CONFIGURAR EL SOCKET CLIENTE Y ENLAZAR LA
COMUNICACIÓN TCP/IP CON EL SERVIDOR

    public class StateObject                              // Objeto que inicializa
    {
       public Socket workSocket = null;                   // Socket en modo cliente.
       public const int BufferSize = 256;                 // Tamaño del buffer.
       public byte[] buffer = new byte[BufferSize];       // Crear nuevo buffer.
       public StringBuilder sb = new StringBuilder();     // Cadena de datos recibida.
    }
    private const int port = 13200;                       //Numero de puerto.
    public static AutoResetEvent connectDone = new AutoResetEvent(false);
    public static AutoResetEvent sendDone = new AutoResetEvent(false);
    public static AutoResetEvent receiveDone = new AutoResetEvent(false);
    private static String response = String.Empty;        // Respuesta del servidor.

    private static void StartClient()                     // Iniciar el cliente.
    {
        try
       {
          IPEndPoint remoteEP = new IPEndPoint(IPAddress.Parse("192.168.0.127"), port);

         Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
         ProtocolType.Tcp);                               // Crea el socket TCP/IP.

          client.BeginConnect(remoteEP, new AsyncCallback(ConnectCallback), client);
          connectDone.WaitOne();                          // Establecer la comunicación con el IPEndPoint.

          Send(client, "<EOF>");                          // Envía la bandera al servidor
          sendDone.WaitOne();

          client.Shutdown(SocketShutdown.Both);
          client.Close();
       }
       catch (Exception e)
       {
         MessageBox.Show (e.ToString());
       }
    }
```

```
private static void ConnectCallback(IAsyncResult ar)
{
   try
   {
      Socket client = (Socket)ar.AsyncState;        // Retrieve the socket from the state object.
      client.EndConnect(ar);                        // Complete the connection.
      connectDone.Set();                            // Signal that the connection has been made.
   }
   catch (Exception e)
   {
            MessageBox.Show (e.ToString());
   }
}

private static void Receive(Socket client)
{
   try
   {
      StateObject state = new StateObject();        // Crea el objeto de estado.
      state.workSocket = client;

      client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new
      AsyncCallback(ReceiveCallback), state);       // Inicia la recepción de datos.
   }
   catch (Exception e)
   {
     MessageBox.Show (e.ToString());
   }
}

private static void ReceiveCallback(IAsyncResult ar)
{
   try
   {
      StateObject state = (StateObject)ar.AsyncState;
      Socket client = state.workSocket;

      int bytesRead = client.EndReceive(ar);            // Lee los datos recibidos.

      if (bytesRead > 0)
      {
         state.sb.Append(Encoding.ASCII.GetString(state.buffer, 0, bytesRead));

         client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new
          AsyncCallback(ReceiveCallback), state);     // Obtiene el resto de los datos.
      }
      else
      {
         if (state.sb.Length > 1)
         {
            response = state.sb.ToString();
         }
```

```
            }
            receiveDone.Set();                          // Señal que se han recibido todos los datos.
        }
        catch (Exception e)
        {
          MessageBox.Show (e.ToString());
        }
    }

    private static void Send(Socket client, String data)
    {
        byte[] byteData = Encoding.ASCII.GetBytes(data);          // convierte a ASCII.
        client.BeginSend(byteData, 0, byteData.Length, 0, new AsyncCallback(SendCallback),
        client);                                                  // Inicia a enviar datos.
    }

    private static void SendCallback(IAsyncResult ar)
    {
        try
        {
            Socket client = (Socket)ar.AsyncState;                // Restablece el socket
            int bytesSent = client.EndSend(ar);                   // Completa el envió de datos
            sendDone.Set();                                       // Señal con todos los bytes
        }
        catch (Exception e)
        {
            MessageBox.Show (e.ToString());
        }
    }
}
```

Anexo C. Código fuente para la aplicación de la Raspberry Pi

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.Threading;
using AForge;
using AForge.Imaging;
using AForge.Imaging.Filters;
using System.Net;
using System.Net.Sockets;
using RaspberryCam;
using System.IO;
using System.Threading.Tasks;
using MathNet.Numerics.LinearAlgebra;


namespace RasPi_App
{
  public partial class Form1 : Form
  {
    private static Cameras camara;
    private List<IntPoint> coordenadas, ideales;
    private double[] theta = new double[8];
    private System.Timers.Timer timer1 = new System.Timers.Timer(4000);
    UnmanagedImage captura, imagen_resultante;
    List<IntPoint> esquinas_cap;
    System.Drawing.Point coordenada;
    public byte umbral = 240;
    private System.Drawing.Point P_actual;
    private System.Drawing.Point coordenada_anterior;
    private Pen lapiz;

    public Form1()
    {
        InitializeComponent();
    }

   private void Form1_Load(object sender, EventArgs e)        // evento cargar formulario
   {
       inicializar();                                          // Inicializa propiedades
       ThreadStart start = new ThreadStart(Inicia_Servidor);   // Inicializa hilo SERVIDOR
       Thread thread = new Thread(start);
       thread.Start();
       CheckForIllegalCrossThreadCalls = false;
    }
```

```
    void inicializar()                                          // Inicializa propiedades
    {
          lapiz = new Pen(Color.Red, 5);                        //Creo el lápiz
          lapiz.StartCap = System.Drawing.Drawing2D.LineCap.Round;
          lapiz.EndCap = System.Drawing.Drawing2D.LineCap.Round;
          lapiz.LineJoin = System.Drawing.Drawing2D.LineJoin.Round;

          ideales = new List<IntPoint>();

          ideales.Add(new IntPoint(0,0));
          ideales.Add(new IntPoint(800,0));
          ideales.Add(new IntPoint(800,600));
          ideales.Add(new IntPoint(0,600));

          esquinas_cap = ideales;
          theta = calcula_homografia(esquinas_cap, ideales);

        //Colocar un picture box encima de otro
        PB1_Screen.Parent = PB2_tablero;  //picture box transparente donde se escribe
        PB1_Screen.Visible = true;              //picture box de fondo, donde se ponen las capturas
        PB1_Screen.Location    =    new    System.Drawing.Point(PB1_Screen.Location.X    -
       PB2_tablero.Location.X, PB2_tablero.Location.Y - PB2_tablero.Location.Y);

        camara = Cameras.DeclareDevice().Named("Camera 1").WithDevicePath
        ("/dev/video0").Memorize();          //Carga el driver de la cámara Raspi Cam

        coordenada_anterior = new System.Drawing.Point(-1, -1);
     }


// evento cerrar formulario
      private void Form1_FormClosed(object sender, FormClosedEventArgs e)
      {
          if (camara.Default != null)
          {
                    if (camara.Default.IsVideoStreamOpenned)
                    {
                              camara.Default.StopVideoStreaming();
                    }
          }
      }


// evento salir del menúToolStrip
      private void salirToolStripMenuItem_Click(object sender, EventArgs e)
      {
          Application.Exit();
      }
```

```
// Codigo correspondiente al A J U S T E DE ESQUINAS
    private void ajustarToolStripMenuItem_Click(object sender, EventArgs e)
    {
        camara.Default.StartVideoStreaming(new PictureSize(1920, 1080), 10);//Streaming on
        timer2.Enabled = true;                                          //Habilita el timer
        button_ajustar.Visible = true;                                  //Visualiza el botón
    }

    private void timer2_Tick(object sender, EventArgs e)                //Tick del timer
    {
        byte[] cap1 = camara.Default.GetVideoFrame();                   //captura un frame
        Stream cap2 = new MemoryStream(cap1);
        System.Drawing.Image cap3 = System.Drawing.Image.FromStream(cap2);
        PB1_Screen.Image = cap3;                                        //Coloca la captura
    }

    private void button_ajustar_Click(object sender, EventArgs e)       //evento del botón
    {
        timer2.Enabled = false;                                         //Deshabilita el timer
        camara.Default.StopVideoStreaming();                            //Streaming off
        byte[] cap = camara.Default.TakePicture(new PictureSize(1920,1080), 10);
        Stream cap2 = new MemoryStream(cap);
        System.Drawing.Image cap3 = System.Drawing.Image.FromStream(cap2);
        PB1_Screen.Image = cap3;
        Form2 ventana_ajustes = new Form2((Bitmap)cap3);    //Habilita la ventana de ajuste
        DialogResult resp = ventana_ajustes.ShowDialog(this);
        if (resp == DialogResult.OK)
        {
                this.esquinas_cap = ventana_ajustes.esquinas_capturadas;
                ventana_ajustes.Dispose();
                PB1_Screen.Image = null;
                theta = calcula_homografia(esquinas_cap, ideales);
         }
         button_ajustar.Visible = false;

    }



// Código correspondiente P R I N C I P A L
    private void detenerToolStripMenuItem_Click(object sender, EventArgs e)//evento stop
    {
        timer1.Enabled = false;                     //deshabilita el timer
        camara.Default.StopVideoStreaming();        //deshabilita el steaming
        if (camara.Default.IsVideoStreamOpenned)    //Verifica que la cámara este apagada
        {
            camara.Default.StopVideoStreaming();
        }
        ajustarToolStripMenuItem.Enabled = true;
    }
```

```
private void IniciarToolStripMenuItem_Click(object sender, EventArgs e) //evento iniciar
{
    ajustarToolStripMenuItem.Enabled = false;
    camara.Default.StartVideoStreaming(new PictureSize(1920, 1080), 10); //Streaming on
    timer1.Elapsed += async (sender2, e2) => await Func();              //Habilita timer async
    timer1.Start();
 }

 private Task Func( )                                                   //función asíncrona del timer
 {
    byte[] capture1 = camara.Default.GetVideoFrame();      //obtiene un frame
    Stream capture2 = new MemoryStream(capture1);
    System.Drawing.Image capture3 = System.Drawing.Image.FromStream(capture2);
    Nueva_Foto((Bitmap)capture3);        //transfiere la captura a una nueva función
    return Task.FromResult<object>(null);
 }

 void Nueva_Foto(Bitmap CapturaCamara)          //Función analizadora de imágenes
 {
    captura = UnmanagedImage.FromManagedImage(CapturaCamara);  //imagen
    Grayscale filtro_gris = new Grayscale(0.2125, 0.7154, 0.0721); // escala de grises
    imagen_resultante = filtro_gris.Apply(captura);// aplica filtro gris a la imagen
    coordenada = lugar_XY(imagen_resultante);    //busca el pixel más brillante en la imagen
    if (coordenada.X == -1)                          // No encontró ningún pixel brillante
    {
       coordenada_anterior = coordenada;
    }
    else                                              // Si se encontró pixel brillante
    {
       P_actual = homografia_punto(theta, coordenada);//aplica homografía al puno pixel

       // Bolque de instrucciones posición actual del puntero al LIRC
            int PosX = P_actual.X;
            int PosY = P_actual.Y;
            this.Cursor = new Cursor(Cursor.Current.Handle);
            Cursor.Position = new System.Drawing.Point(PosX,PosX);
       //Termina posicionamiento del puntero

       if (coordenada_anterior.X != -1)              // Existía un pixel brillante anterior
       {
            MDibujar(coordenada_anterior, P_actual); // Dibuja línea entre puntos
       }
       else                  // Es el primer pixel brillante valido por lo que grafica un punto
       {
            MDibujar(P_actual, P_actual); // Dibuja un punto únicamente
       }
       coordenada_anterior = P_actual;
    }

 }
```

```
System.Drawing.Point lugar_XY(UnmanagedImage senal) // LOCALIZA EL PIXEL MÁS BRILLANTE

{
    for (int x = 0; x < imagen_resultante.Width; x++)     //Barrido horizontal de la imagen
    {
       for (int y = 0; y < imagen_resultante.Height; y++)//Barrido vertical de la imagen
       {
          Color pixelColor = imagen_resultante.GetPixel(x, y);  //Obtiene el color del pixel
          if (pixelColor.R >= umbral)                  //compara el color con el umbral
          {
             coordenada = new System.Drawing.Point(x, y); //Se encontró pixel brillante
             return coordenada;                          //Regresa la coordenada
          }

       }
    }
    return coordenada = new System.Drawing.Point(-1, -1);
}

List<System.Drawing.Point> Lista = new List<System.Drawing.Point>();   // DIBUJAR

public void MDibujar( System.Drawing.Point s)
{
     Graphics g1 = PB1_Screen.CreateGraphics();                  //Crea lienzo de dibujo
     g1.DrawRectangle(lapiz,new Rectangle((int)s.X, (int)s.Y, 3,3)); //Dibuja Puntos
     Lista.Add(new System.Drawing.Point(s.X, s.Y));              //Guarda los puntos en la lista

    if (Lista.Count > 1)                                    //Obtiene los puntos
    {
            g1.DrawLines(lapiz, Lista.ToArray());          //Traza una línea entre los puntos
            g1.Dispose();
    }
}

private void button1_Click(object sender, EventArgs e) //EVENTO DE LA GOMA
{
     PB1_Screen.Refresh();                                  //Limpia el lienzo
}

private double[] calcula_homografia(List<IntPoint> P_XY, List<IntPoint> P_IJ)
{                                       // CALCULA LA HOMOGRAFÍA DE LAS ESQUINAS
 double[] v_theta = new double[8];
 double[] B ={ P_IJ[0].X, P_IJ[0].Y, P_IJ[1].X, P_IJ[1].Y, P_IJ[2].X, P_IJ[2].Y, P_IJ[3].X, P_IJ[3].Y };
 double[,]A ={
            { P_XY[0].X , P_XY[0].Y, 1, 0, 0, 0, -(P_IJ[0].X * P_XY[0].X), -(P_IJ[0].X * P_XY[0].Y) },
            { 0, 0, 0, P_XY[0].X , P_XY[0].Y, 1, -(P_IJ[0].Y * P_XY[0].X), -(P_IJ[0].Y * P_XY[0].Y) },
            { P_XY[1].X , P_XY[1].Y, 1, 0, 0, 0, -(P_IJ[1].X * P_XY[1].X), -(P_IJ[1].X * P_XY[1].Y) },
            { 0, 0, 0, P_XY[1].X , P_XY[1].Y, 1, -(P_IJ[1].Y * P_XY[1].X), -(P_IJ[1].Y * P_XY[1].Y) },
            { P_XY[2].X , P_XY[2].Y, 1, 0, 0, 0, -(P_IJ[2].X * P_XY[2].X), -(P_IJ[2].X * P_XY[2].Y) },
            { 0, 0, 0, P_XY[2].X , P_XY[2].Y, 1, -(P_IJ[2].Y * P_XY[2].X), -(P_IJ[2].Y * P_XY[2].Y) },
            { P_XY[3].X , P_XY[3].Y, 1, 0, 0, 0, -(P_IJ[3].X * P_XY[3].X), -(P_IJ[3].X * P_XY[3].Y) },
            { 0, 0, 0, P_XY[3].X , P_XY[3].Y, 1, -(P_IJ[3].Y * P_XY[3].X), -(P_IJ[3].Y * P_XY[3].Y) },
             };
```

```
            Matrix a = Matrix.Create(A);
            Matrix b = new Matrix(B,8);
            Matrix x = a.Solve(b);

            for (int i = 0; i < 8; i++)
            {
                        v_theta[i] = x[i, 0];
            }
            return v_theta;
    }
     private System.Drawing.Point homografia_punto(double[] theta, System.Drawing.Point P)
    {                                       // DIBUJA LA HOMOGRAFÍA DE UNA COORDENADA
            System.Drawing.Point transformado = new System.Drawing.Point();
            transformado.X = (int)((theta[0] * P.X + theta[1] * P.Y + theta[2]) / (theta[6] * P.X + theta[7] * P.Y + 1));
            transformado.Y = (int)((theta[3] * P.X + theta[4] * P.Y + theta[5]) / (theta[6] * P.X + theta[7] * P.Y + 1));
            return transformado;
    }


//Código principal del SERVIDOR
      private static bool activo = true;
      private static int puerto = 13200; // Determina el puerto de comunicación con el cliente
      private static byte[] buffer;
      private static Socket Sck_Server, Sck_Cliente;
      public static ManualResetEvent allDone;

      public void Inicia_Servidor()                     // Función de inicialización del socket
      {
         Console.CancelKeyPress += new ConsoleCancelEventHandler(Console_CancelKeyPress);
         allDone = new ManualResetEvent(false);
         try
         {
          Sck_Server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
           Sck_Server.Bind(new IPEndPoint(IPAddress.Any, puerto));
           Sck_Server.Listen(110);
           while (activo)
           {
               allDone.Reset();
               Sck_Server.BeginAccept(new AsyncCallback(AcceptCallback), Sck_Server);
               allDone.WaitOne();
           }
         }
         catch (Exception ex)
         {
            MessageBox.Show(ex.Message);
         }
         finally
         {
            if (Sck_Server != null)
            {
               Sck_Server.Close(100);
```

```
        }
        if (Sck_Cliente != null)
        {
            Sck_Cliente.Close();
        }
    }
}

public void AcceptCallback(IAsyncResult ar)          //Aceptar conexiones del cliente
{
    try
    {
        allDone.Set();
        if (activo)
        {
            Sck_Cliente = Sck_Server.EndAccept(ar);
            buffer = new byte[256];
            Sck_Cliente.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new
            AsyncCallback(ReceiveCallback), null);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error AceptCallback: {0}", ex.Message);
    }
}

public void ReceiveCallback(IAsyncResult ar)              //Recibe los datos del cliente
{
    try
    {
        int num = Sck_Cliente.EndReceive(ar);
        if (num > 1)
        {
            try
            {
                 Bitmap captura = new Bitmap("/home/pi/Desktop/Capreta
                CompartidaPI/ScreenShot.png");
                PB2_tablero.Image = captura; // Carga la imagen en el plano de proyección
            }
            catch (Exception)
            {
                MessageBox.Show("Error AceptCallback: {0}", ex.Message);
            }
            Sck_Cliente.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new
            AsyncCallback(ReceiveCallback), null);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Excepcion", ex.Message.ToString());
    }
```

```
        }

        private static void Console_CancelKeyPress(object sender, ConsoleCancelEventArgs e)
        {
            e.Cancel = true;
            Form1.activo = false;
            try
            {
                IPEndPoint remoteEP = new IPEndPoint(IPAddress.Loopback, Form1.puerto);
                Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                ProtocolType.Tcp);
                socket.Connect(remoteEP);
                socket.Send(new byte[1]);
                socket.Close();
            }
            catch (Exception ex)
            {
               MessageBox.Show(ex.Message);
            }
        }

    }
}

//CÓDIGO DE LA VENTANA AJUSTE DE ESQUINAS
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using AForge;
using AForge.Imaging;

namespace Ras Pizarrón
{
    public partial class Form2 : Form
    {
        private byte no_click = 0;
        public List<IntPoint> esquinas_capturadas = new List<IntPoint>();
        private Bitmap capturado;
        public Form2(Bitmap imagen)
        {
            InitializeComponent();
            capturado = imagen;
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            pictureBox1.Image = capturado;
            esquinas_capturadas.Clear();
        }

        private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
```

```
        {
            toolStripStatusLabel1.Text = "X: " + e.X.ToString() + " Y: " + e.Y.ToString();
        }

        private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
        {
            Graphics puntero = pictureBox1.CreateGraphics();
            Pen pluma = new Pen(Brushes.Red,1);

            if (no_click == 0)
            {
                this.label_P1.Text = "X1: " + e.X.ToString() + "   Y1: " + e.Y.ToString();
                esquinas_capturadas.Add(new IntPoint(e.X, e.Y));
                puntero.DrawEllipse(pluma, e.X - 10, e.Y - 10, 20, 20);
                puntero.DrawLine(pluma, e.X - 10, e.Y, e.X + 10, e.Y);
                puntero.DrawLine(pluma, e.X, e.Y-10, e.X, e.Y+10);
            }
            else if (no_click == 1)
            {
                this.label_P2.Text = "X2: " + e.X.ToString() + "   Y2: " + e.Y.ToString();
                esquinas_capturadas.Add(new IntPoint(e.X, e.Y));
                puntero.DrawEllipse(pluma, e.X - 10, e.Y - 10, 20, 20);
                puntero.DrawLine(pluma, e.X - 10, e.Y, e.X + 10, e.Y);
                puntero.DrawLine(pluma, e.X, e.Y - 10, e.X, e.Y + 10);
            }
            else if (no_click == 2)
            {
                this.label_P3.Text = "X3: " + e.X.ToString() + "   Y3: " + e.Y.ToString();
                esquinas_capturadas.Add(new IntPoint(e.X, e.Y));
                puntero.DrawEllipse(pluma, e.X - 10, e.Y - 10, 20, 20);
                puntero.DrawLine(pluma, e.X - 10, e.Y, e.X + 10, e.Y);
                puntero.DrawLine(pluma, e.X, e.Y - 10, e.X, e.Y + 10);
            }
            else if (no_click == 3)
            {
                this.label_P4.Text = "X4: " + e.X.ToString() + "   Y4: " + e.Y.ToString();
                esquinas_capturadas.Add(new IntPoint(e.X, e.Y));
                button1.Enabled = true;
                puntero.DrawEllipse(pluma, e.X - 10, e.Y - 10, 20, 20);
                puntero.DrawLine(pluma, e.X - 10, e.Y, e.X + 10, e.Y);
                puntero.DrawLine(pluma, e.X, e.Y - 10, e.X, e.Y + 10);
            }
            no_click++;
            pluma.Dispose();
            puntero.Dispose();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Referencias

[1]. Borja Fernandez, Vico. Archivo Digital UPM. [En línea] 16 de julio de 2015. http://oa.upm.es/40003/8/PFC_BORJA_FERNANDEZ_VICO.pdf.

[2]. Gómez García, Soledad. *Conceptos y Funcionalidas basícas.* [En línea] 2010. https://cefire.edu.gva.es/pluginfile.php/277763/mod_resource/content/2/Unidad1/unidad1_html/index.html.

[3]. Arduino. *The Arduino Mini05.* [En línea] 2017. https://www.arduino.cc/en/Guide/ArduinoMini#connecting.

[4]. Wikipedia. *Processing.* [En línea] https://es.wikipedia.org/wiki/Processing.

[5]. —. *Wiring.* [En línea] https://es.wikipedia.org/wiki/Wiring.

[6]. Freescale. *Proximity Capacitive Touch Sensor Controller* . [En línea] 2010. https://www.sparkfun.com/datasheets/Components/MPR121.pdf.

[7]. Semiconductors, NXP. *I2C-bus specification and user manual.* [En línea] UM10204.

[8]. Ada, Lady. *Introducing the Raspberry Pi Model B+.* [En línea] 14 de 03 de 2015. https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-model-b-plus-plus-differences-vs-model-b.pdf.

[9]. Foundation, Raspberry Pi. *Installing Operating System Images.* [En línea] 2017.

[10]. Debian. [En línea] https://www.debian.org/index.es.html.

[11]. Bartelmus, Karsten Scheibler & Christoph. LIRC. [En línea] 26 de Mayo de 2016. http://www.lirc.org/.

[12]. Camera Module . [En línea] Raspberry Pi Foundation, 2013. https://www.raspberrypi.org/documentation/hardware/camera/README.md.

[13]. Stone, Ash. *The Raspberry Pi Camera.* The MagPi. [En línea] 2013. https://www.raspberrypi.org/magpi/.

[14]. Philips RC5 Infrared Transmission Protocol . [En línea] Philips , 13 de Sep de 2017. **http://techdocs.altium.com/display/FPGA/Philips+RC5+Infrared+Transmission+Protocol.**

[15]. Sony SIRC Infrared protocol. [En línea] SONY, Febrero de 2010. http://picprojects.org.uk/projects/sirc/sonysirc.pdf.

[16]. Gonzalez, R. & Woods .R. *Digital Image Processing.* **New Jersey : Prentice Hall, 2002. Second Edition.**

[17]. Melo, Samuel Barreto. *TRANSFORMACIONES GEOMÉTRICAS SOBRE IMÁGENES DIGITALES***.** Colombia : Universidad Distrital Francisco José de Caldas , 2009.

[18]. Calvo, Nestor. *Transformaciones.* Argentina : Universidad Nacional del Litoral, 2007.

[19]. Foley, James D. *Introducción a la graficación por computador.* Buenos Aires : Addison-Wesley Iberoamericana, 1996.

[20]. Microsoft. *Introduction to the C# Language and the .NET Framework.* [En línea] 20 de Julio de 2015. https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework.

[21]. Mono. C# Compiler. *Mono.* [En línea] 2017. http://www.mono-project.com/docs/about-mono/languages/csharp/.

[22]. Instructables. *Burn a New Bootloader - Arduino Pro Mini.* [En línea] 25 de Septiembre de 2016. http://www.instructables.com/id/Burn-a-New-Bootloader-Arduino-Pro-Mini/.

[23]. Inc., Prolific Technology. *PL-2303 Datasheet.* Taiwan : s.n., 2005.

[24]. Arduino. *Wire Library.* [En línea] https://www.arduino.cc/en/Reference/Wire.

[25]. NaylampMechatronics. *Libreria IR remote.* [En línea] 25 de Agosto de 2016. http://www.naylampmechatronics.com/blog/36_Tutorial-Arduino-y-control-remoto-Infrarrojo.html.

[26]. Systems, Advanced Monolithic. *AMS1117 Datasheet.* [En línea] Advanced Monolithic Systems . http://www.advanced-monolithic.com/pdf/ds1117.pdf.

[27]. Microsoft. *Programación orientada a eventos.* [En línea] https://msdn.microsoft.com/es-es/library/edzehd2t(v=vs.110).aspx.

[28]. —. *GetAsyncKeyState function***. [**En línea] https://msdn.microsoft.com/en-us/library/windows/desktop/ms646293(v=vs.85).aspx.

[29]. —. *Bitmap class.* [En línea] https://msdn.microsoft.com/en-us/library/system.drawing.bitmap(v=vs.110).aspx.

[30]. Alliance, Wi-Fi. *Wi-Fi Alliance* . [En línea] https://www.wi-fi.org/discover-wi-fi.

[31]. Microsoft. *Sockets asincrónos.* [En línea] https://msdn.microsoft.com/es-es/library/fx6588te(v=vs.110).aspx.

[32]. —. *Clase Thread* . [En línea] https://msdn.microsoft.com/es-es/library/system.threading.thread(v=vs.110).aspx.

[33]. Sharpe, Richard. *Samba.* [En línea] https://www.samba.org/cifs/docs/what-is-smb.html**.**

[34]. Flechner, Romain. *RaspberryCam.* [En línea] https://bitbucket.org/rflechner/raspberrycam/wiki/Home.

[35]. Microsoft. *Programación asíncrona en .NET.* [En línea] 26 de mayo de 2017. https://msdn.microsoft.com/es-es/communitydocs/net-dev/dev/programacion-asincrona-en-net.