Universidad Autónoma de Yucatán

Facultad de Matemáticas

Automatización del laberinto elevado en forma de cruz mediante sensores capacitivos para la medición de ansiedad en ratas

TESIS

Presentada por:

I.C. Emmanuel de Jesús Vera Tun

En opción al título:

Maestro en Ciencias de la Computación

Asesores:

Dr. Francisco José Heredia López

Dr. José Luis Góngora Alfaro

Mérida, Yucatán, México

2017

LA AMISTAD ES UN BUEN ATAJO PARA UN CAMINO LARGO Y DÍFICIL.

DEDICATORIA

El presente trabajo se lo dedico a mis padres, hermanos y amigos que me apoyaron en esta tesis, en especial a mi hermano Marvin Ulises Vera Tun, que gracias a su apoyo pude llegar hasta este punto en el que estoy y tener una formación profesional.

AGRADECIMIENTOS

Agradezco a mi familia, tanto a mis padres y a mis hermanos que me brindaron su apoyo para poder realizar la maestría.

Agradezco al Dr. Francisco José Heredia López y al Dr. José Luis Góngora Alfaro por darme la oportunidad de realizar esta tesis y por transmitirme una gran cantidad de conocimientos.

Al CIR “Dr. Hideyo Noguchi” por financiar el proyecto, al CONACYT por otorgarme una beca para estudiar la maestría y a la Facultad de Matemáticas de la UADY por aprobar la realización de este trabajo, sobre todo al Dr. Fernando Curi Quintal, coordinador de la maestría en ciencias de la computación, que me aconsejo y facilitó la toma de decisiones administrativas.

A la compañía Azoteq que me brindó asesoría técnica para poder utilizar sus dispositivos.

Agradezco especialmente a Paula Cen Brito, la persona que fue mi motivación para que yo tomará la decisión de realizar la maestría. Su apoyo y motivación fueron fundamentales al inicio de la maestría.

Principalmente, agradezco a las siguientes personas que contribuyeron en este trabajo, tanto técnica y emocionalmente, brindándome sus conocimientos, habilidades, apoyo personal y sobre todo su gran amistad cuando más lo necesité: Cindy Marisol Espadas Aguilar, Jorge Koyoc Koyoc, Anakaren Ruiz Tamayo, Patricio Montes de Oca, Higinio Pool Tzuc y Daniel Chacón Matú. El aporte de estas personas fue fundamental para poder concluir este trabajo.

RESUMEN

El laberinto elevado en forma de cruz es un equipo empleado para llevar a cabo experimentos relacionados con la ansiedad en ratas; consta de una plataforma de acrílico en forma de cruz elevada a 50 cm del suelo, que cuenta con dos brazos con paredes y dos sin paredes. Al inicio del experimento la rata es colocada en el centro de la cruz, y durante 5 minutos se registra el número de veces que el animal entra en cada brazo y el tiempo que permaneció en cada uno de estos. Por instinto los roedores suelen visitar más veces y pasar mayor tiempo en los brazos cerrados, ya que en ellos encuentran protección contra posibles amenazas. Por lo anterior, este método es útil para medir el nivel de ansiedad del roedor y se emplea para identificar fármacos que disminuyen la ansiedad (ansiolíticos) ya que estos aumentan las entradas y el tiempo de permanencia en los brazos abiertos. Sin embargo, durante la experimentación, la presencia de la persona que registra la conducta de la rata puede alterar su comportamiento, alterando los resultados. Para evitar lo anterior, usualmente se graba el experimento con una videocámara para que posteriormente un observador analice la filmación para contabilizar las entradas y tiempos de permanencia de la rata en los brazos abiertos y cerrados. Sin embargo, lo anterior está sujeto a fallas que pueden ser generadas por distracciones humanas. Para evitar estos conflictos, en el presente proyecto se planteó el objetivo de automatizar el registro de la posición del roedor en el laberinto elevado en forma de cruz utilizando sensores capacitivos colocados debajo de la plataforma a fin de evitar perturbaciones en el comportamiento del roedor que puedan alterar los resultados. El sistema que se implementó es capaz de registrar el estado de los sensores en una memoria interna y también transferir esta información en un archivo en formato JSON a una memoria USB, por medio de comunicación bluetooth a un teléfono móvil y a una base de datos mediante comunicación WiFi. Posteriormente, el archivo obtenido es procesado mediante un algoritmo implementado en LabVIEW®, de tal manera que se obtenga información útil para el experimentador. También cuenta con una batería de respaldo para seguir funcionando en caso de un corte de energía. Durante la evaluación experimental el sistema fue capaz de registrar la presencia física de una rata adulta; sin embargo, en su estado actual el dispositivo no pudo detectar la posición correcta del roedor en cada segundo

del experimento, pues hubieron conflictos relacionados con la topología de comunicación en la lectura de los sensores, dando como resultado que no se obtuviera información suficiente para ubicar con precisión la posición del roedor a lo largo del tiempo. Como trabajo futuro se planea resolver este conflicto modificando la topología de comunicación diseñada en este trabajo, y permitiendo así, una lectura adecuada sin pérdida de información de los sensores a lo largo del experimento.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

1. INTRODUCCIÓN

La ansiedad es un estado de tensión o nerviosismo aumentado, y también puede entenderse como una respuesta fisiológica del cerebro ante situaciones de peligro potencial que activa al eje hipotálamo-pituitaria-adrenal, promoviendo la liberación de diversas hormonas (ACTH, cortisol, adrenalina y noradrenalina) que modifican las funciones orgánicas con el propósito de que el individuo responda eficientemente para la lucha o el escape (Graeff, 2007; Schneiderman, Ironson, & Siegel, 2005). La ansiedad moderada puede contribuir a incrementar la atención y la concentración; sin embargo, la ansiedad crónica puede dar lugar a un exceso en la liberación de glucocorticoides, y a un deterioro en la función cognitiva (de Kloet, Oitzl, & Joëls, 1999; McEwen & Sapolsky, 1995).

Cuando los animales son sometidos a estrés, estos generan una mezcla compleja de respuestas fisiológicas y de comportamiento. Aunque esta respuesta de estrés es fundamental para la supervivencia, si se prolonga por periodos largos puede ocasionar deterioro de la salud física y mental, dando lugar a desórdenes afectivos tal como depresión y muchas enfermedades sistémicas y neurodegenerativas (Schneiderman et al., 2005). Los desórdenes de ansiedad son comunes en la población general y están frecuentemente asociados con la depresión (Schneiderman et al., 2005).

El deseo de mejorar en las actuales terapias ansiolíticas ha estimulado las investigaciones farmacológicas preclínicas empleando modelos de ansiedad en roedores, los cuales se pueden agrupar en dos clases: modelos “etológicos”, en los que se mide la respuesta de animales normales expuestos a situaciones ambientales desconocidas; y los modelos de “entrenamiento” en los que se mide la respuesta del animal a un castigo (Sahgal, 1993). Las pruebas de ansiedad con animales son usadas para identificar nuevos medicamentos con actividad ansiolítica, investigar la neurobiología de la ansiedad, y evaluar el impacto de otras situaciones que generan estrés, tal como el olor de un predador, o primeras experiencias de crianza (Davis, 2001).

Entre los diversos métodos para medir la ansiedad en animales, el laberinto elevado en forma de cruz ha ganado popularidad por su sencillez de ejecución (Walf & Frye, 2007). Esta prueba depende del análisis de comportamiento de los roedores expuestos al conflicto de explorar un área desconocida y evadir los espacios que representan un riesgo. Puede ser usado en ratas o ratones de cualquier sexo (Davis, 2001).

Figura 1: Vista de un laberinto elevado en cruz que nos ayudará a comprender mejor su forma.

El método del laberinto elevado en forma de cruz fue realizado como una modificación de los trabajos del Dr. A. M. J. Montgomery en su artículo “The relation between fear induced by novel stimulation and exploratory behavior”, publicado en el Journal of Comparative and Physiological Psychology en 1958. Esta modificación fue realizada de tal manera que se diseñó un laberinto con cuatro brazos (dos abiertos y dos cerrados) que están organizados en forma de una cruz, tal como podemos observar en la Figura 1. De acuerdo a los estudios de Montgomery, en los brazos abiertos se

crea un fuerte enfoque de evasión, mientras que en los brazos cerrados esto no sucede (Lister, 1987). De la misma forma, se ha demostrado que existe una marcada reducción en el tiempo dedicado a recorrer los brazos abiertos, en comparación con el empleado en permanecer en los brazos cerrados (Walf & Frye, 2007). Esta conducta está relacionada con la aversión innata que los roedores muestran por las áreas elevadas y desprotegidas (Gonzalez & File, 1997).

El registro de la conducta del animal (número de entradas y tiempos de permanencia en brazos abiertos y cerrados) es llevado a cabo por diferentes técnicas, siendo la más común que un observador lleve el registro durante el experimento. Sin embargo, la presencia de la persona puede alterar el comportamiento del roedor, sesgando los resultados (Torres & Escarabajal, 2002). Para evitar lo anterior, una alternativa es que el experimentador grabe en video todo el experimento, para luego reproducirlo en cámara lenta y así poder contar el número y la duración de las visitas a cada tipo de brazo del laberinto. De esta manera el experimentador podrá reproducir el video cuantas veces sea necesario para medir con precisión estos parámetros, aunque siempre existe la posibilidad de fallas humanas debidas a distracción o cansancio, especialmente cuando se analizan muchos experimentos (Torres & Escarabajal, 2002).

Una solución para evitar los conflictos señalados es el uso de sistemas electrónicos que contabilizan de forma automatizada las entradas y tiempos de permanencia en los brazos del laberinto elevado en forma de cruz. Actualmente ya se han publicado varios trabajos describiendo dichos sistemas, los cuales han sido validados para el estudio de la ansiedad en roedores.

2. ANTECEDENTES

2.1. Métodos del registro de laberintos automatizados

2.1.1. Celdas Fotoeléctricas

Torres & Escarabajal (2002) describieron un método que emplea 10 pares de celdas fotoeléctricas estratégicamente colocadas en varias partes del laberinto. En la Figura 2 se muestra un esquema de la distribución de los sensores, los cuales permiten detectar la presencia del roedor en los brazos del laberinto.

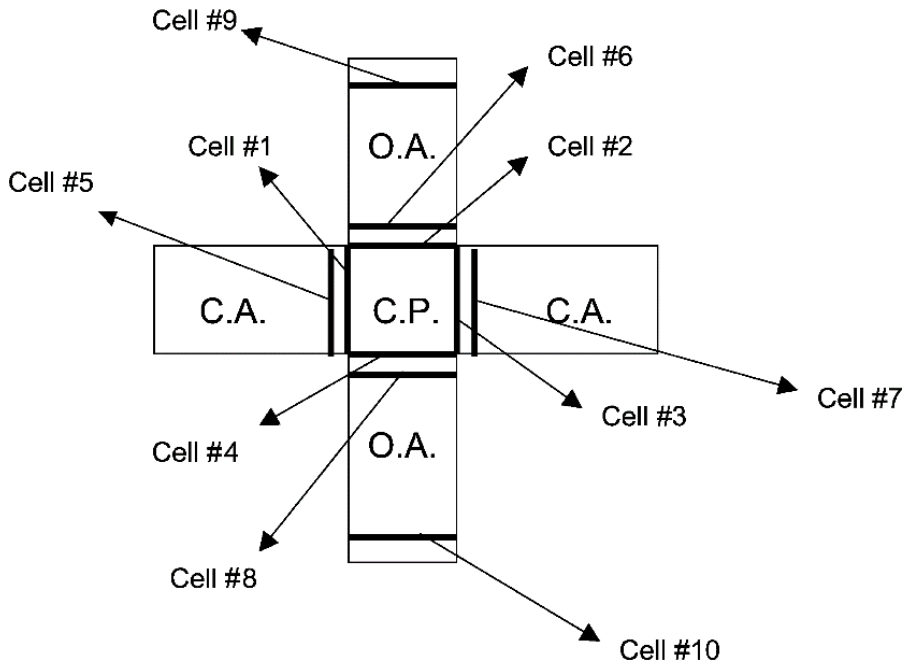


Figura 2: Laberinto elevado en forma de cruz automatizado con celdas fotoeléctricas [imagen tomada de Torres & Escarabajal, 2002]

En el mercado se pueden encontrar laberintos automatizados que utilizan la tecnología de celdas fotoeléctricas (Panlab Technology for Bioresearch, s/f). Sin embargo, este sistema tiene la desventaja de que, para cada experimento, es necesario ajustar manualmente las posiciones de los sensores que indican la entrada del roedor a un

brazo (celdas #5, #6, #7 y #8) de acuerdo al tamaño del roedor que se vaya a evaluar, ya que el algoritmo utilizado para la detección de entradas del animal a los brazos se basa en el cambio de luz que ocurre cuando el roedor se encuentra posicionado entre los sensores que rodean la zona central (celdas #1, #2, #3 y #4) y los que indican entradas a los brazos.

2.1.2. Sensores infrarrojos

Otro sistema, ilustrado en la Figura 3, utiliza sensores infrarrojos para detectar al roedor sobre el laberinto (Sidor, Rilett, & Foster, 2010). A pesar de ser un sistema bastante sencillo y económico, la estabilidad del funcionamiento de este tipo de sensores depende mucho de la alineación y distancia entre su emisor y receptor infrarrojo, y de las condiciones de iluminación en el ambiente, por lo que pequeños desajustes o luces brillantes pueden hacer que el sistema no funcione adecuadamente.

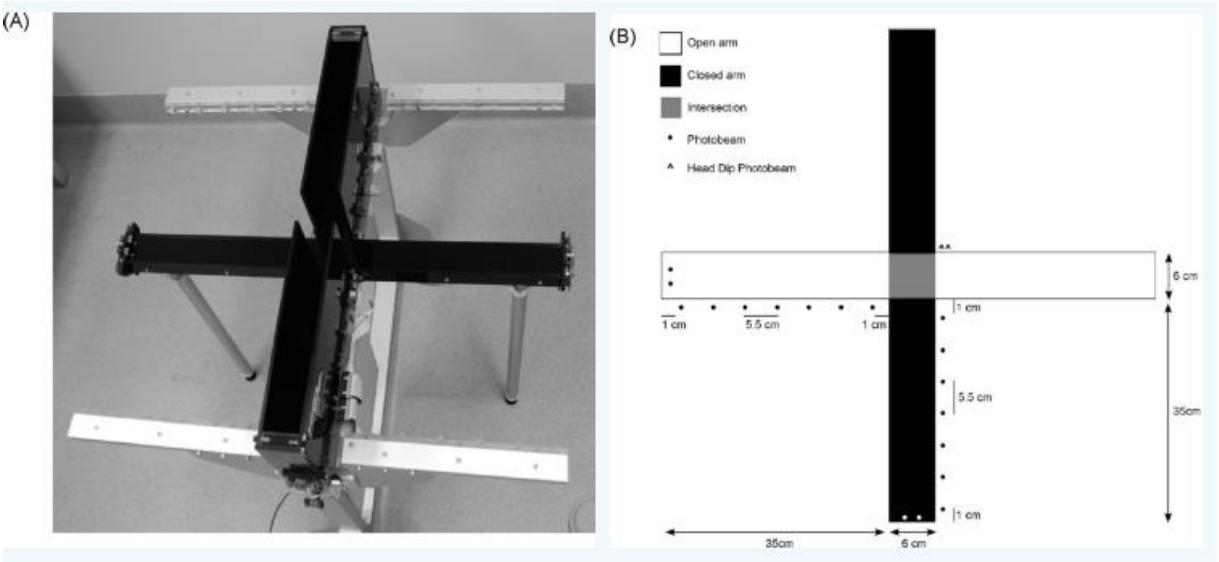


Figura 3: Sistema del laberinto elevado en forma de cruz automatizado con sensores infrarrojos: (A) muestra una foto del laberinto, y (B) muestra un esquema de la posición de los sensores. [imagen tomada de Sidor et al, 2010]

2.1.3. Análisis de imágenes

Otro método se basa en la filmación de la trayectoria del roedor y el análisis computacional de las imágenes utilizando el contraste entre el color del animal y el color del piso del laberinto, el cual se divide en 5 secciones: cuatro brazos y la intersección entre ellos (Patel, Seasholtz, & Patel, 2006). En el mercado ya se pueden adquirir sistemas que funcionan por medio de visión computacional y que cuentan con software que mide ciertos parámetros, como tiempo de permanencia y porcentaje de visitas en cada brazo, latencia para salir del primer brazo, entre otros (Biobserve, s/f; Campden Instruments Ltd, s/f). Este sistema puede observarse en la Figura 4.

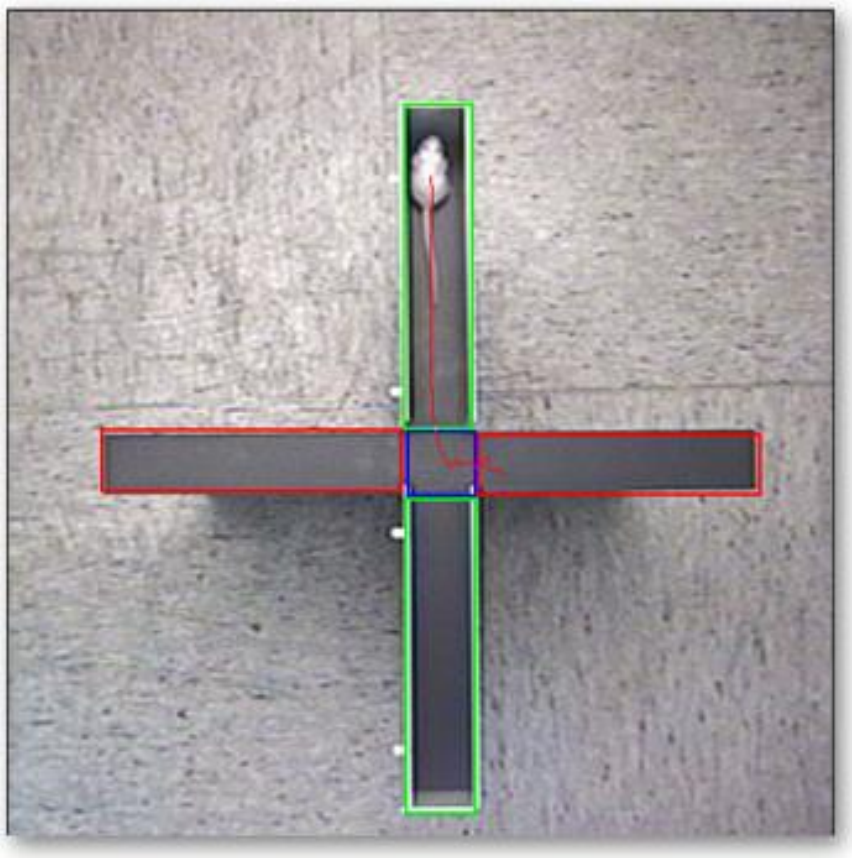


Figura 4: Sistema automatizado por visión computacional (Harvard Apparatus, s/f)

Sin embargo, los precios de estos equipos son excesivos, y pueden estar fuera del alcance de laboratorios que realizan diversos experimentos de conducta además del laberinto elevado en forma de cruz.

2.2. Sensado capacitivo como método alterno para el registro en laberintos automatizados

El avance de la tecnología ha permitido contar con dispositivos táctiles capacitivos de bajo costo y suficientemente sensibles para detectar y ubicar la posición de partes corporales, como los dedos una mano, que análogamente pueden aplicarse a la detección de las patas de un roedor. Es por ello que en el presente proyecto se planteó el objetivo de usar esta tecnología para el desarrollo de un sistema automático para monitorear con precisión las diferentes entradas a los brazos y ubicar la posición de roedores en el laberinto elevado en forma de cruz.

2.2.1. Diseño preliminar de un laberinto elevado en forma de cruz con sensores capacitivos

En el laboratorio de Ingeniería Biomédica del CIR “Dr. Hideyo Noguchi” de la UADY se desarrolló un prototipo electrónico para detectar la ubicación de un roedor en los brazos y zona central del laberinto elevado en forma de cruz, colocando sensores capacitivos debajo de la plataforma de acrílico, de tal forma que la posición del animal pueda ser detectada sin que éste se percate de los sensores ubicados debajo (Vera, Heredia, Góngora, Bata, & Álvarez, 2013). En la Figura 5 se muestra la ubicación de estos sensores en el laberinto en forma de cruz (áreas de color rojo).

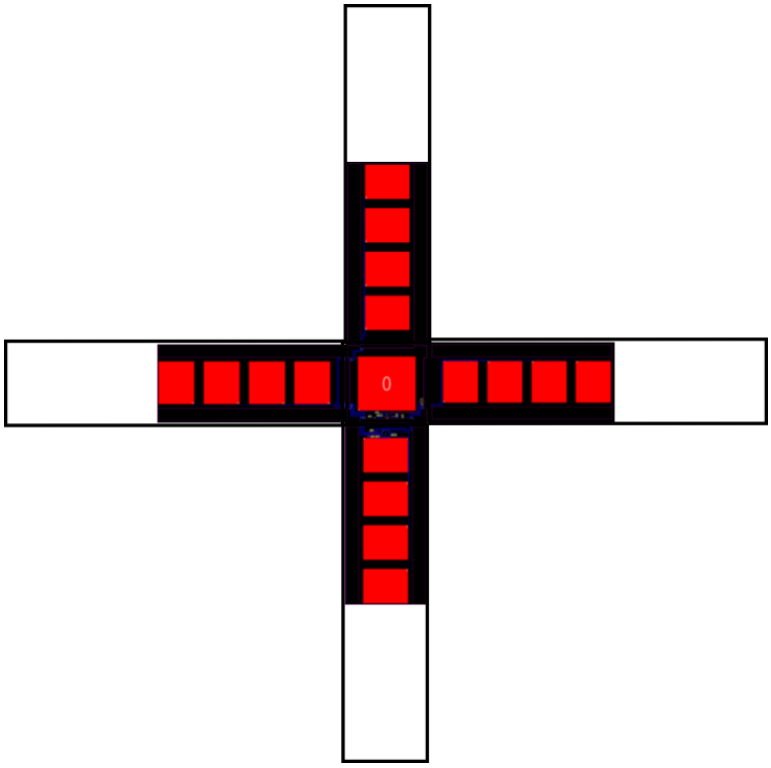


Figura 5: Esquema de la distribución de los sensores capacitivos en el laberinto elevado en forma de cruz.

En cada uno de los brazos del laberinto se colocaron cuatro sensores de 6 x 5 cm y la zona central tiene uno de 7.8 x 7.9 cm. Las dimensiones de los sensores se eligieron para detectar el cuerpo completo de una rata adulta (250-300 g). La distribución de los sensores en cada brazo abarcó la mitad adyacente a la zona central del laberinto, lo cual fue suficiente para detectar que la rata logró meter las cuatro patas dentro del brazo, que es el criterio usualmente empleado por observadores humanos que analizan la conducta de los roedores en el laberinto en forma de cruz.

El sistema se equipó con un microcontrolador PIC16LF727 de Microchip®, el cual cuenta internamente con los circuitos necesarios para generar un oscilador con el sensor capacitivo que sea colocado en alguna de sus entradas especializadas. Este chip tiene la capacidad de monitorear hasta 16 sensores capacitivos utilizando la tecnología *self-capacitance*.

Para lograr el monitoreo de cada sensor, se requiere medir la frecuencia de la señal del sensor. Para obtener dicho valor, el microcontrolador usa dos temporizadores internos: uno para contar las cargas y descargas de voltaje del sensor y el segundo para programar un tiempo en el que se realizan estas cuentas. El sistema incorpora un contador de lecturas para leer varias veces la frecuencia del sensor y realizar un promedio de éstas. Primero el sistema mide la frecuencia varias veces cuando los sensores están desocupados hasta tener un promedio de esta frecuencia. Posteriormente continua con la medición de la frecuencia de cada sensor; por cada lectura de frecuencia que se obtenga del sensor se realiza una resta con la frecuencia promedio, y el resultado es comparado con un valor de umbral definido previamente, que es un porcentaje de la frecuencia promedio, para considerar que lo que se detectó es parte del cuerpo de la rata. El algoritmo se ilustra en detalle en la Figura 6.

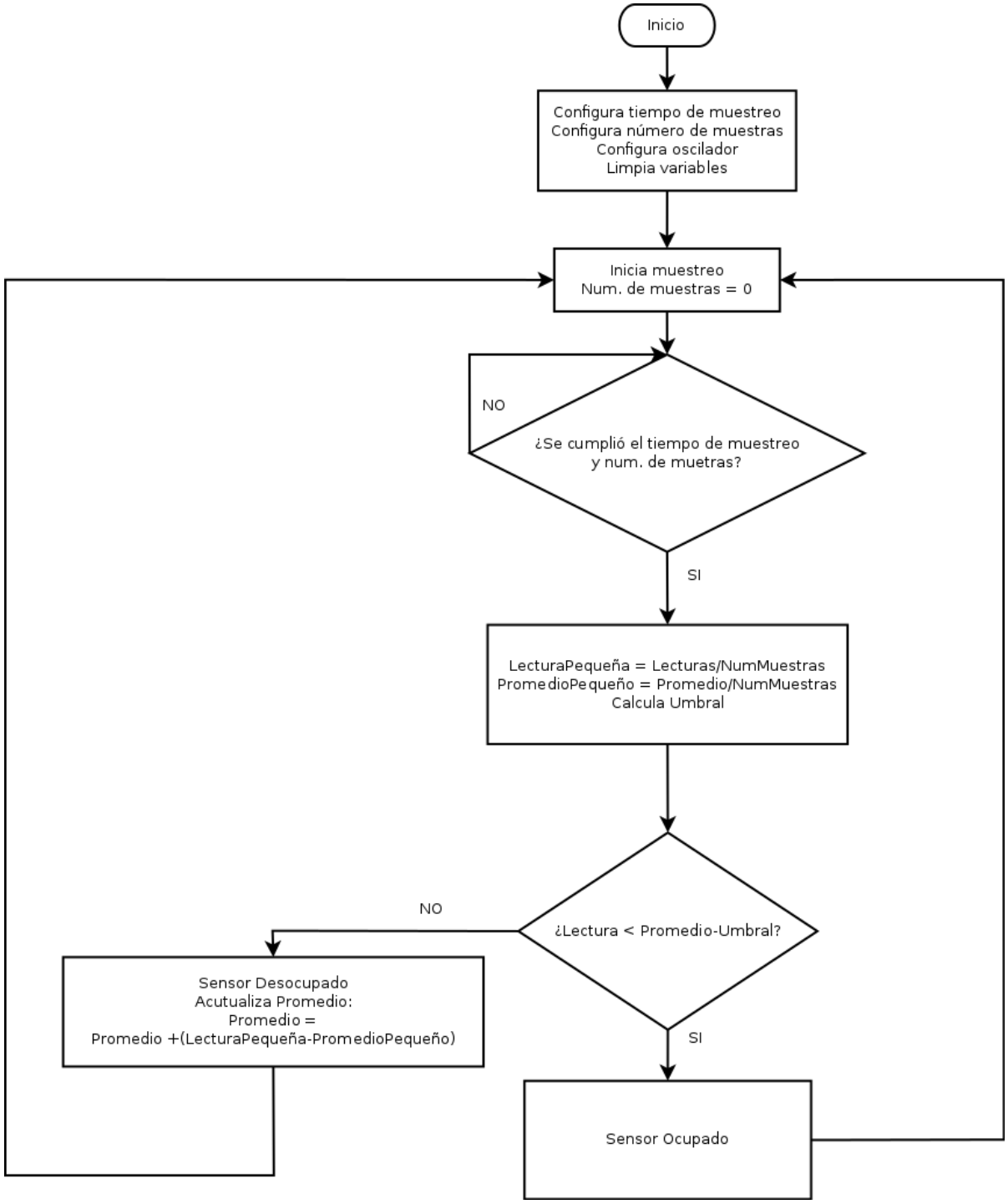


Figura 6: Diagrama de flujo que describe el algoritmo usado para determinar si un sensor capacitivo está ocupado o desocupado

Sin embargo, durante la evaluación experimental de este prototipo se detectó que, debido al tamaño y arreglos de los sensores que utiliza, éstos son activados por la cola

del animal, por lo que los cambios de postura que realiza el roedor durante el experimento dan lugar a que el sistema detecte falsas entradas en los brazos del laberinto. Debido a lo anterior, en el presente proyecto se plantea una alternativa para lograr que el sistema de sensado capacitivo del laberinto elevado en forma de cruz detecte la posición de la rata con la misma precisión que el método tradicional de captura y análisis por vídeo.

2.2.2. Descripción del laberinto elevado en forma de cruz a automatizar con sensado capacitivo de precisión

El laberinto que se automatizará en este proyecto consta de una base de madera en forma de cruz, sostenida por 4 patas de madera a una altura de 50 cm del piso. Sobre esta base se asienta una placa de acrílico de 0.5 cm de espesor y con forma de cruz con las siguientes dimensiones: cuatro brazos de 50 cm de largo x 11 cm de ancho, que convergen en una zona central cuadrada de 11 x 11 cm (Figura 7).

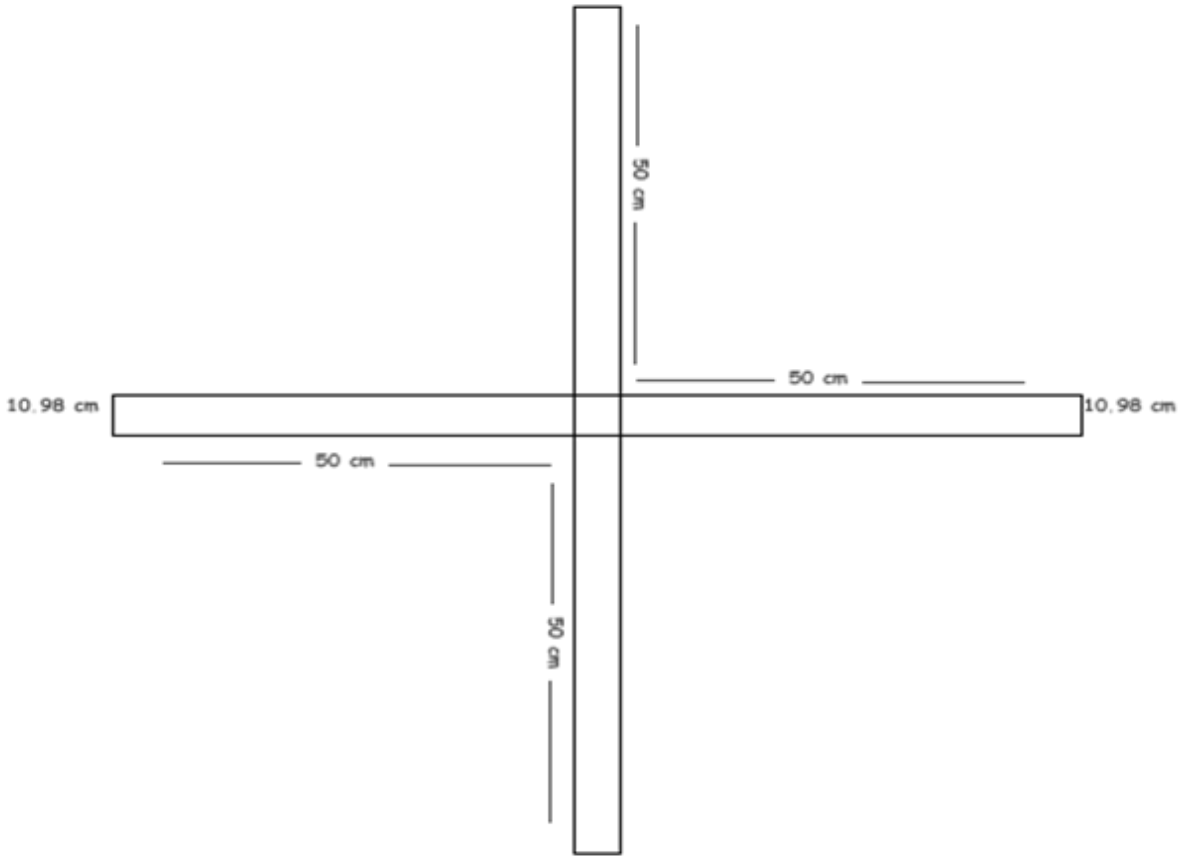


Figura 7: Esquema del laberinto elevado en cruz, vista superior.

Como se muestra en la Figura 8, dos de los brazos cuentan con paredes de 40 cm de altura, y los otros dos brazos carecen de paredes.

Figura 8: Vista completa del laberinto elevado en cruz, así como su base.

3. HIPÓTESIS

Un sistema de detección de posición basado en una matriz de sensores capacitivos de pequeñas dimensiones y controlados por microcontroladores “Reduced Instruction Set Computer” (RISC) que realicen procesamiento distribuido, es un método económico para ubicar con mayor precisión la posición de una rata en el laberinto elevado en forma de cruz.

4. OBJETIVO GENERAL

Automatizar el laberinto elevado en forma de cruz con una matriz de sensores capacitivos de pequeñas dimensiones, con el fin de crear un sistema para capturar la posición del animal en experimentación, ubicar y analizar su posición con precisión dentro del laberinto.

4.1. Objetivos específicos

- Diseñar sensores capacitivos de pequeñas dimensiones que, dispuestos en forma de matriz debajo la plataforma de acrílico del laberinto, permitan detectar con precisión la ubicación de una rata.
- Diseñar e implementar circuitos electrónicos y programas para microcontrolador que interpreten las señales recibidas por los sensores, las almacenen en la memoria una vez interpretadas y permitan su transferencia a una memoria flash USB, a un teléfono móvil mediante comunicación Bluetooth® o a una base de datos remota por medio de Wi-Fi.
- Implementar un programa para computadora que analice los datos de posición del roedor capturados por el sistema, los tiempos de permanencia en los brazos abiertos, cerrados y en la zona central.
- Elaborar la documentación técnica y de operación correspondiente.

5. MARCO TEÓRICO

5.1. Tecnologías de sensado capacitivo

Actualmente los sensores capacitivos pueden ser construidos directamente sobre una tarjeta de circuito impreso con la forma y distribución física más adecuada para la aplicación que se requiera. Además de ser de bajo costo, tienen una excelente sensibilidad para tareas de detección. Por otra parte, tienen algunas limitaciones de detección cuando se usan guantes u otro tipo de aislantes que se interpongan entre el objeto y el sensor.

Algunas de las técnicas básicas para procesar la información obtenida del sensor capacitivo están basadas en la medición de la frecuencia o ciclo de operación de la señal generada por el sensor, que genera un cambio debido a la introducción de capacitancia adicional con respecto un punto de referencia, por ejemplo, entre el dedo de una persona y el suelo (Perme, 2007a).

Para que un sensor capacitivo pueda detectar la presencia de un objeto, se deben realizar las siguientes tareas:

Crear un circuito oscilador que generará una señal periódica con el sensor capacitivo.

Medir varias veces la frecuencia de la señal generada en ausencia de objetos cerca del sensor capacitivo. Esto con el fin de obtener una frecuencia promedio correspondiente a un sensor desocupado, es decir sin presencia del objeto a detectar.

Medir la frecuencia de la señal en presencia de un objeto cerca del sensor capacitivo.

Restar las frecuencias obtenidas anteriormente, y determinar si la diferencia es significativa para considerar que el sensor está realmente ocupado.

Para lograr lo anterior se requiere de un sistema capaz de ejecutar este algoritmo, lo cual puede ser realizado por medio de un microcontrolador o mediante un sistema implementado en un circuito integrado (Perme, 2007c). El circuito oscilador que se requiere para un sensor capacitivo se puede construir a partir de un simple circuito como el ilustrado en la Figura 9

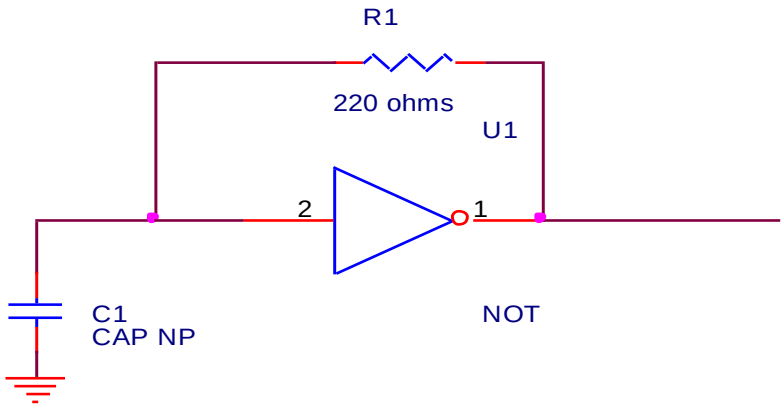


Figura 9: Circuito oscilador simple con compuerta lógica negadora

Este circuito realiza las labores que hemos visto con anterioridad; es decir, genera un oscilador por medio del condensador C1 (sensor capacitivo), el cual se cargará y descargará gracias a la retroalimentación de la compuerta lógica negadora, ya que inicialmente C1 tendrá un voltaje inicial que será interpretado como un “0” o “1” lógico por la compuerta. Este valor se invierte a la salida de esta compuerta, por lo que, si el valor de entrada es “0”, indicando C1 descargado, su salida será “1”, es decir, un voltaje alto, por lo que proporcionará un voltaje a C1 para que comience a cargarse. Mientras C1 se está cargando, su voltaje incrementará hasta que la compuerta lo consideré un “1” lógico en su entrada, indicando C1 cargado, esto provocará que el valor de salida de la compuerta se convierta en un “0” lógico, un voltaje bajo, por lo que C1 comenzará a reducir su carga hasta que su voltaje se interprete como un “0” lógico nuevamente. Esto hará que la frecuencia de la señal osciladora dependa directamente de la capacidad del condensador para almacenar cargas (capacitancia), haciendo sensible de esta manera el cambio de frecuencia por capacitancias parásitas,

que es la característica más importante para poder detectar la presencia de un objeto cercano.

El cuerpo humano tiene su propia capacitancia que esta referenciada a tierra física, lo que ocasiona que al momento de aproximarse al condensador del circuito de la Figura 9 (ubicado físicamente en la zona que se interesa monitorear), se sumen las capacitancias del sensor y la parásita, haciendo que la frecuencia disminuya considerablemente debido al incremento de la capacitancia total del circuito. Por lo que midiendo la frecuencia del oscilador podemos saber si un objeto está cercano o tocando la zona donde se ubica el sensor (Perme, 2007b).

Con la llegada de las pantallas táctiles capacitivas en la vida diaria, se ha ido incrementando su demanda en el mercado, como por ejemplo los teléfonos móviles, cajeros automáticos, trackpads, etc. Con el fin de lograr una superficie táctil para este tipo de aplicaciones, se requiere que la superficie cuente con varios puntos sensibles distribuidos sobre toda la superficie de interés a ser monitoreada (Akhtar, Kakarala, & Member, 2014).

Con el fin de lograr la detección más precisa y óptima de objetos sobre superficies, la tecnología de sensado capacitivo ha ido evolucionando y actualmente podemos encontrar dos arquitecturas diferentes para su implementación: self-capacitance y mutual-capacitance.

5.1.1. Self-capacitance

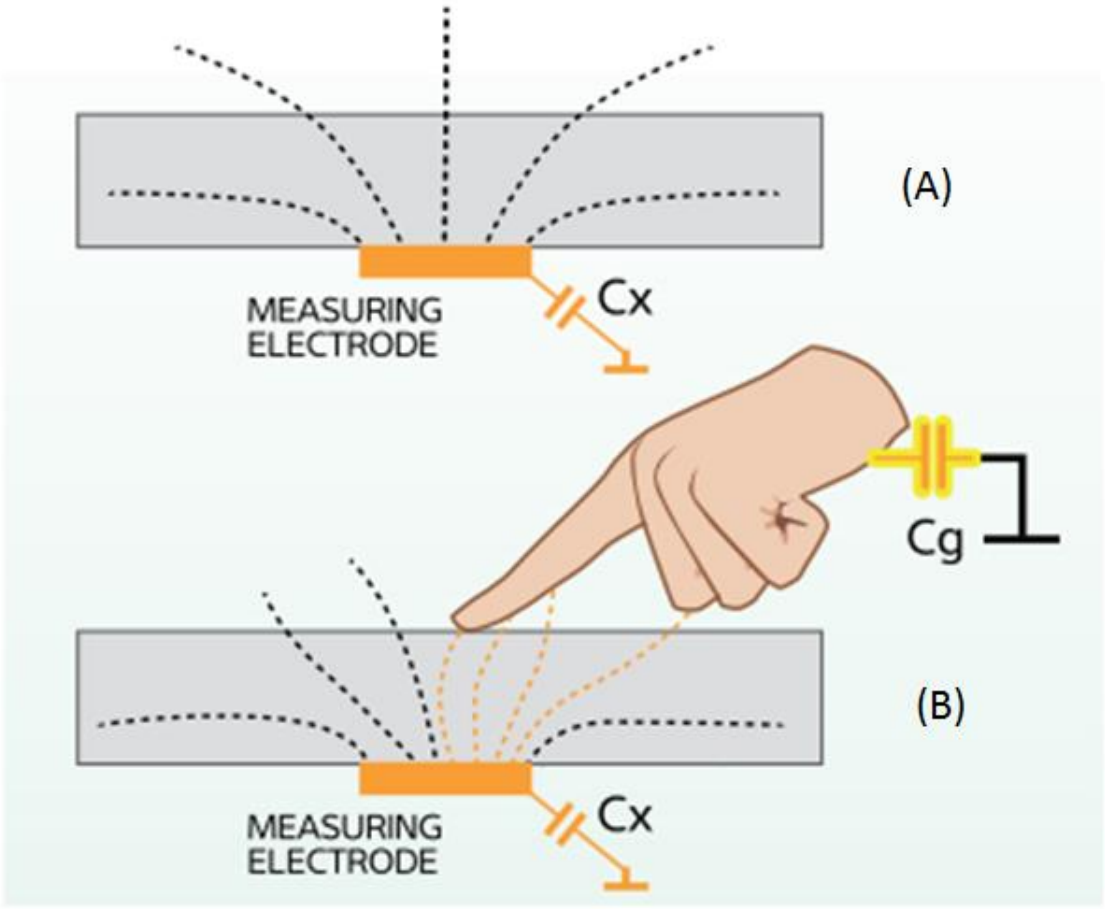


Figura 10: Topología de un electrodo self-capacitance (A) sin capacitancia parásita y (B) y con capacitancia parásita [imagen tomada de (Hristov, 2011)*]*

Con la tecnología *self-capacitance*, cada sensor consta de un solo electrodo en el que se construye un circuito como el de la Figura 9. En la Figura 10 podemos observar como físicamente el sensor de esta topología es construido para detectar objetos. Una técnica para ubicar la posición de un objeto en un área usando la tecnología *self-capacitance*, es distribuir los electrodos en forma de matriz, en donde cada fila está conformada por un solo electrodo, al igual que cada columna (Figura 11). Si un objeto, con capacitancia suficiente para activar un sensor, es colocado en la intersección de

una fila y columna, se activará todo el sensor fila y todo el sensor columna, y con esta información es posible conocer la posición del objeto en el arreglo de sensores. Sin embargo, una de las limitantes de esta tecnología es que no es posible determinar la posición de dos o más objetos que son colocados en diferentes intersecciones. Por ejemplo, en la Figura 12 se muestra que dos objetos son colocados en las intersecciones (X2,Y0) y (X1,Y3), pero esto provocará que se activen los 4 sensores: X1, X2, Y0 y Y3, por lo que no es posible determinar que no hay objetos en las intersecciones (X1,Y0) y (X2,Y3); a estas intersecciones se les llama puntos fantasma (*ghost points*) (Barrett & Omote, 2010).

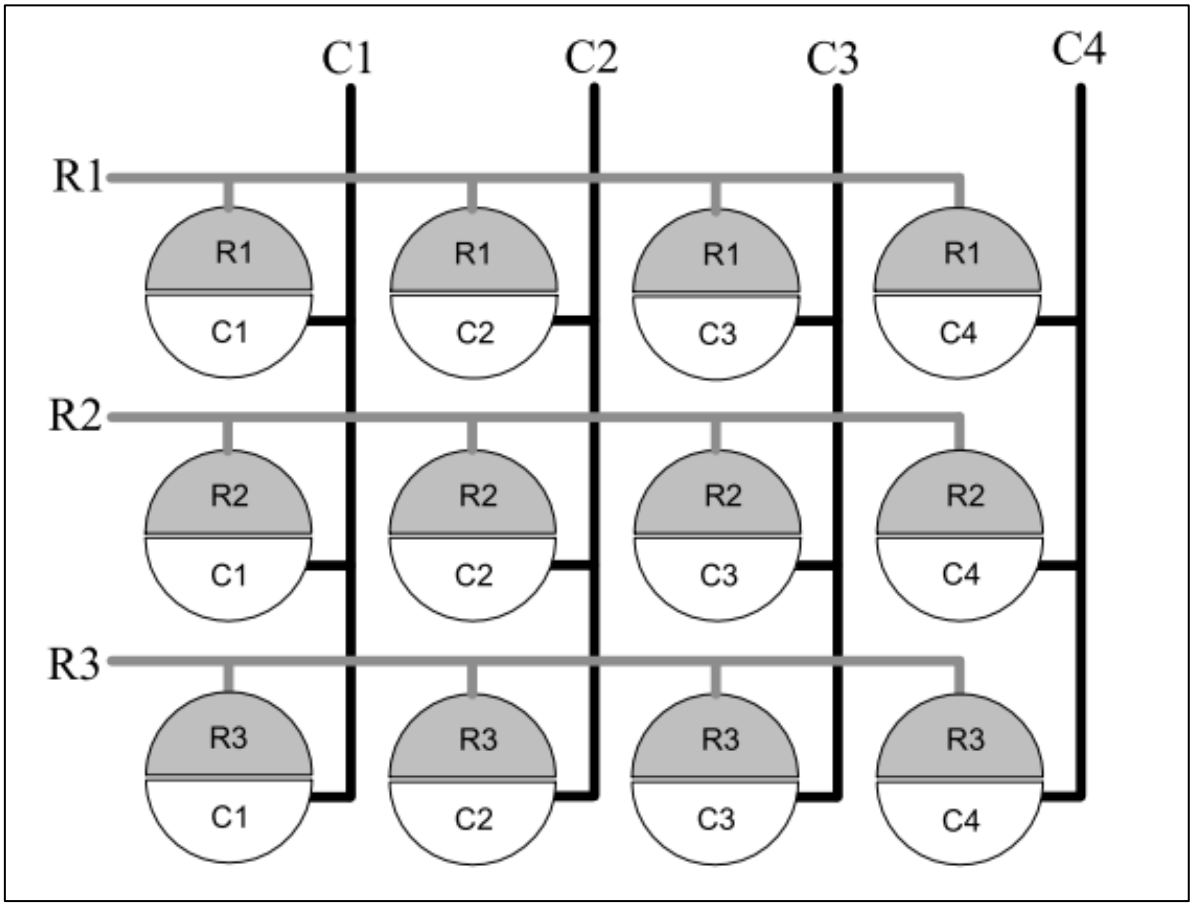


Figura 11: Arreglo de sensores con arquitectura self-capacitance [imagen tomada de (Bohn, 2009)*].*

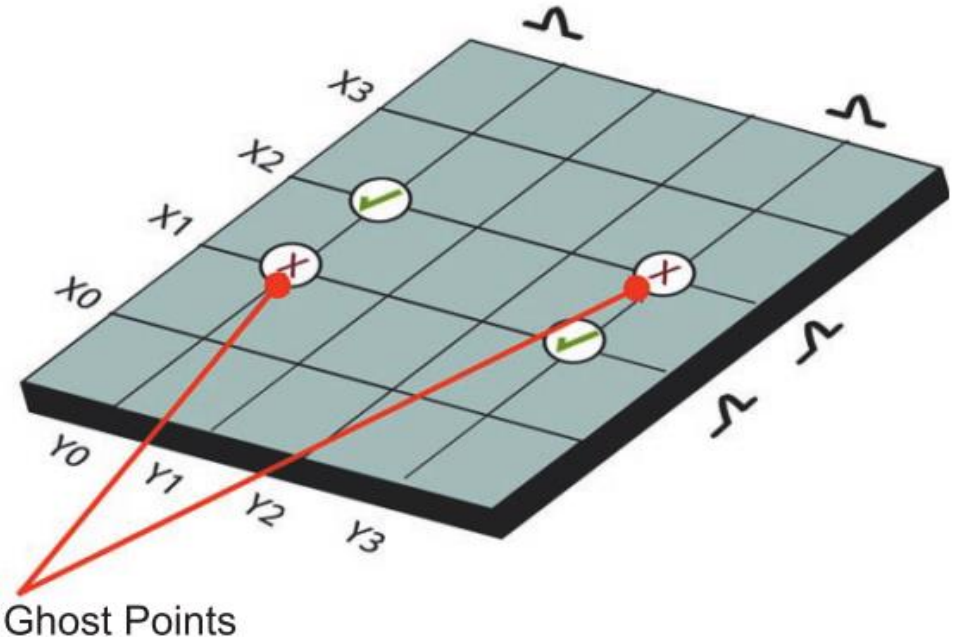


Figura 12: Problema con tecnología self-capacitance [imagen tomada de (Barrett & Omote, 2010)*]*

5.1.2. Mutual-capacitance

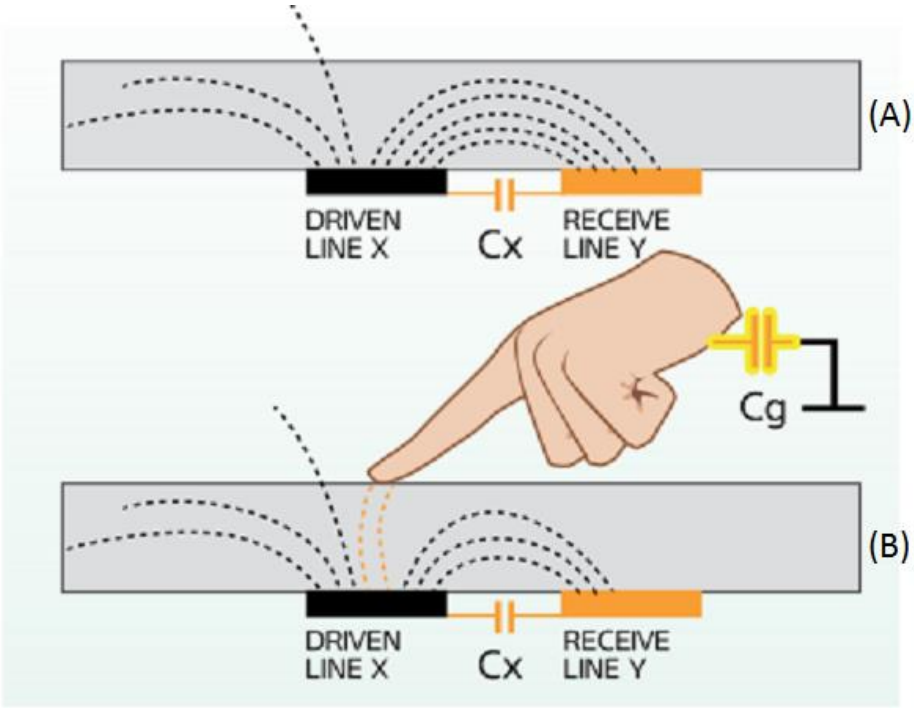


Figura 13: Topología de un canal mutual-capacitance (A) sin capacitancia parásita y (B) con capacitancia parásita [imagen tomada de (Hristov, 2011)*]*

Con la tecnología *mutual-capacitance* (o de capacitancia proyectada) cada sensor (también llamado canal) está formado por dos electrodos, uno emisor y otro receptor (Figura 13). El emisor genera una señal osciladora y el campo eléctrico producido es captado por el receptor. Cuando se interpone un objeto con capacitancia entre los dos electrodos, cambia significativamente la frecuencia que obtiene el receptor y el sistema determina si el canal está ocupado. Los canales también pueden ser organizados en forma de matriz, de tal manera que todas las filas sean emisoras y todas las columnas receptoras, o viceversa. Para determinar si un canal está ocupado, la tecnología *mutual-capacitance* activa únicamente el emisor de esa fila o columna mientras lee todos los receptores, permitiendo determinar de esta manera cuál está ocupado (Figura 14). En el ejemplo de la imagen, donde las X's son transmisores y las Y's receptores, dos objetos se encuentran en la superficie, uno ubicado en la posición (X2, Y0) y el otro en la posición (X1,Y3); cuando el transmisor X2 está activado y los demás deshabilitados, el único receptor que detecta algún objeto es Y0; ocurre lo mismo cuando X2 está transmitiendo, sólo el receptor Y3 es capaz de detectar el cambio. Dado que el diseño de esta tecnología permite determinar qué canal o canales están interrumpidos, es posible determinar con precisión el número de objetos y la ubicación de cada uno en toda la superficie táctil, lo cual tiene mayor ventaja que el método de *self-capacitance* (Barrett & Omote, 2010).

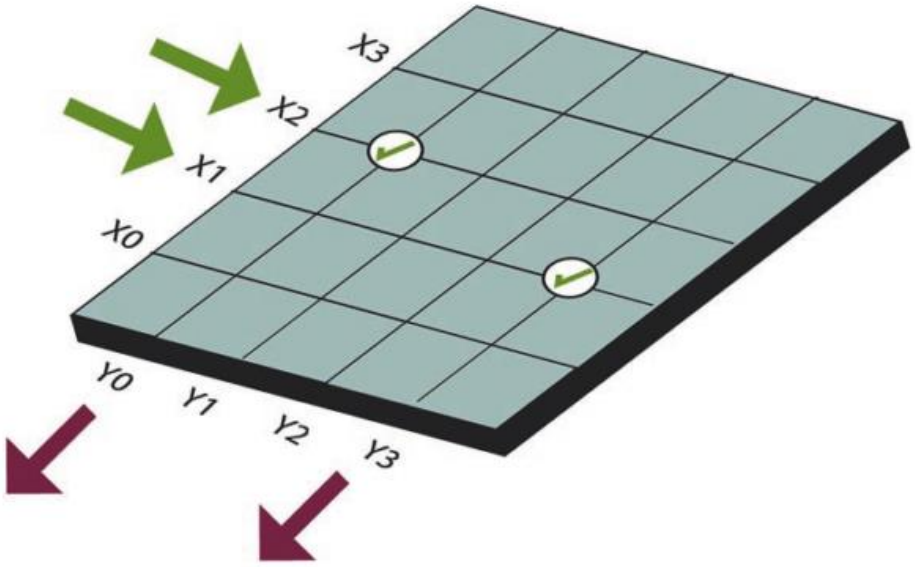


Figura 14: Con la tecnología mutual-capacitance es posible determinar de manera simultánea las posiciones de varios objetos en una superficie [imagen tomada de (Barrett & Omote, 2010)*]*

5.2. Criterios de diseño de sensores capacitivos

Los sensores capacitivos usualmente son áreas rellenas de metal colocados en una placa de circuito impreso; sin embargo, las de mayor uso se encuentran en pantallas táctiles y están construidos con óxido de estaño de indio (ITO). En el presente documento nos enfocaremos en los sensores construidos en circuitos impresos.

En la Figura 15 se puede observar los componentes principales que se deben de tomar en cuenta al diseñar un sensor capacitivo: panel frontal, el sensor (*mutual* o *self*), circuito impreso (PCB) y los trazos de tierra. Para todos estos componentes es importante tomar en cuenta diferentes propiedades de cada uno de ellos, como por ejemplo el espesor, material, distancias entre cada uno, áreas, etc.

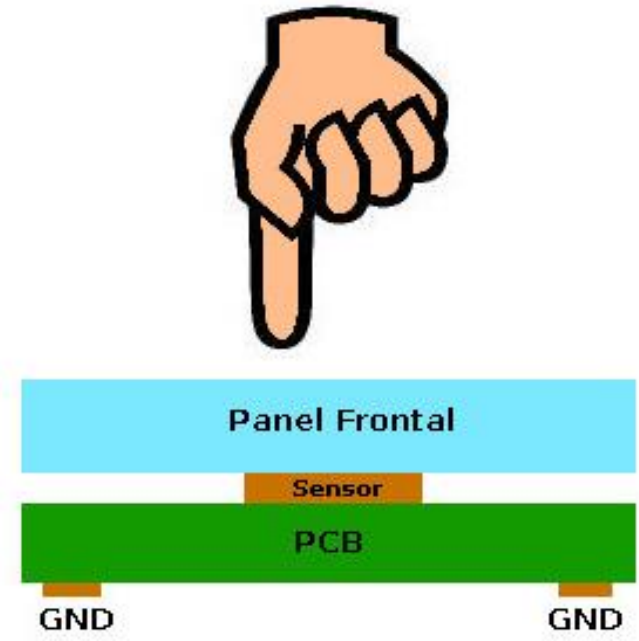


Figura 15: Componentes principales de un sensor capacitivo

En la bibliografía podemos encontrar diferentes diseños de sensores, que difieren en forma, tamaño y distribución sobre la plataforma que se desea monitorear. Los diseños se definen a partir de la aplicación que se requiera y generalmente cada fabricante de circuitos especializados en sensado capacitivo maneja sus propios criterios de diseños a partir de pruebas experimentales (por ejemplo, Azoteq®, Microchip®, Atmel®, entre otros). Usualmente podemos encontrar tres enfoques en el diseño de sensores capacitivos basados en su aplicación, botones táctiles, proximidad y aplicaciones en la cual se requiere detectar la posición de un objeto, gestos, número de objetos presentes, etc. Éste último enfoque tiene mayor aplicación para el diseño de *trackpads* (como el control del ratón en las laptops) y pantallas táctiles.

A partir del esquema ilustrado en la Figura 15 podemos definir la siguiente terminología:

- C_x: Capacitancia natural del electrodo, libre de cualquier capacitancia parásita.
- ε_r: Constante dieléctrica relativa del panel frontal.

- ε_0: Capacitancia por metro del espacio libre, definida como $8.85 \times 10^{-12} F/m$.
- T: Espesor del panel frontal en metros.
- A: Área de la región que fue tocada por un objeto (m^2).
- C_p: Cualquier capacitancia parásita extra, en paralelo con C_x.
- SNR: Relación señal-ruido: una medida de calidad de la medición de la capacitancia.

La capacitancia se define como en la ecuación (1).

$$C = \frac{\varepsilon_0 \times \varepsilon_r \times A}{T} \tag{1}$$

Por lo que claramente podemos notar que si el panel frontal es delgado y tiene una constante dieléctrica muy alta, se logra un gran cambio de capacitancia durante el toque de un objeto y por consecuente una ganancia alta una mejor SNR (Atmel, 2011).

En el contexto de los sensores capacitivos, SNR está definida como la ecuación (2) y la Figura 16 complementa dicha ecuación para una mejor comprensión.

$$SNR(dB) = 20Log\left(\frac{IntensidadDeToque}{RuidoDeToque_{RMS100}}\right)$$

$$IntensidadDeToque = SeñalConToque_{AVG100} - SeñalSinToque_{AVG100}$$

$$RuidoDeToque = \sqrt{\frac{\sum_{n=0}^{99}(Señal[n] - SeñalConToque_{AVG100})^2}{100}} \tag{2}$$

Donde:

AVG100 es el promedio de 100 puntos de datos, típicamente tomados igualmente espaciados sobre un periodo de dos segundos.

RMS100 es la media cuadrática de 100 puntos de datos, típicamente tomados igualmente espaciados sobre un periodo de dos segundos, usando la figura de AVG100 como una basal.

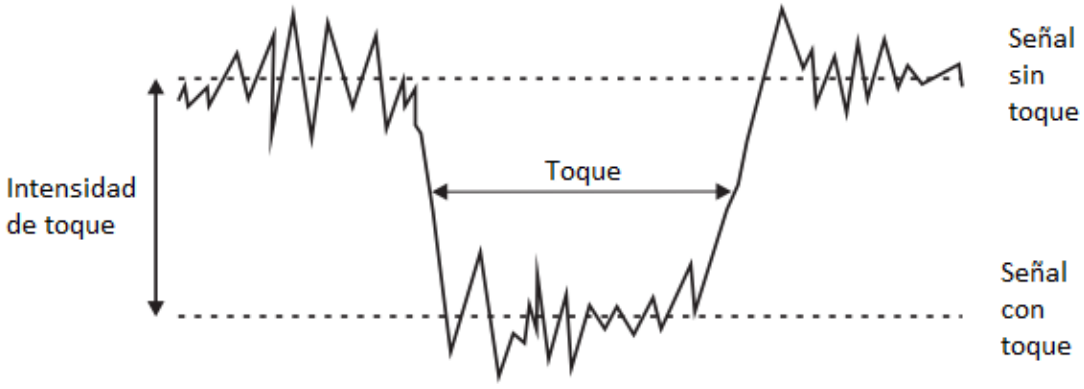


Figura 16: Niveles de señales de toques [imagen tomada de (Atmel, 2011)*]*

Otra de los factores que hay que tomar en cuenta es el sustrato, que es el material en donde se encuentran construidos los sensores. Cualquier material puede ser usado como sustrato, pero los sustratos de baja pérdida son generalmente preferibles, tales como materiales de PCB (FR4, CEM-1, Poliamida y Kapton por nombrar unos cuantos), acrílicos como tereftalato de polietileno (PET) o policarbonato (PC). El cristal también es un excelente material (Atmel, 2011).

El material del sensor o electrodo (cobre, carbón, tinta de plata, ITO, etc) mientras menor es la resistividad Ω/sq del material hace más fácil el manejo de la constante de tiempo RC del sensor. La elección del rango de Ω/sq está íntimamente ligado con la forma y el tamaño del electrodo.

El espesor del panel frontal y su constante dieléctrica (ε_r) son un parte muy importante para determinar la intensidad del campo eléctrico en la superficie del panel. El cristal tiene una ε_r más alta que la mayoría de los plásticos. Altos valores implican que el campo se propagará a través del panel más eficientemente. Por ejemplo, un panel de 5 mm de grosor con una ε_r de 8 tendrá una sensibilidad similar a la de un panel 2.5

mm con una ε_r de 4. Mientras más grueso es un material, la SNR es peor. Por tal razón siempre es mejor intentar y reducir el grosor del material del panel frontal. También es preferible usar materiales con una constante dieléctrica muy alta para ayudar a incrementar la SNR. En la Tabla 1 podemos observar una lista de las constantes dieléctricas de materiales comúnmente utilizado es este tipo de aplicaciones.

Tabla 1: Materiales con sus respectivas constantes dieléctricas [tabla tomada de (Atmel, 2011)*]*

Material	**Constante dieléctrica (ε_r)**
Vacío	1
Aire	1.00059
Vidrio	3.7 a 10
Vidrio de zafiro	9 a 11
Mica	4 a 8
Nylon	3
Silicio	11 a 12
Goma de silicona	3.2 a 9.8
Compuesto de moldeo de silicona	3.7
Papel	2
Plexiglás	3.4
Policarbonato	2.9 a 3
Polietileno	2.2 a 2.4
Poliestireno	2.56
PET	3

Vidrío pírex	4.3 a 5
Cuarzo	4.2 a 4.4
Goma	3
FR4 (Fibra de vidrio + epóxico)	4.2
PMMA (Polimetil metacrilato)	2.6 a 4

Una medida útil para seleccionar de un conjunto de paneles de diferentes grosores y materiales es el factor de sensibilidad (S), que está dada en la ecuación (3).

$$S = \frac{\varepsilon_r}{t} \tag{3}$$

Donde:

t: Es el grosor del panel

En caso de tener un conjunto de paneles de diferentes materiales apilados, la constante combinada está dada por la ecuación (4).

$$\frac{1}{S_{STACK}} = SUM\left(\frac{1}{S_{LAYER}[n]}\right) \tag{4}$$

Donde:

n: es el número de capas.

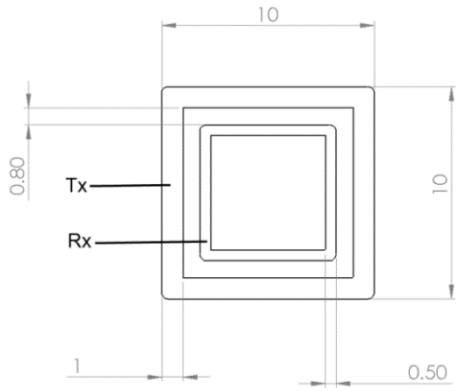
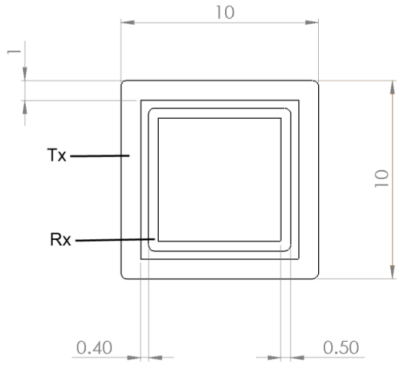
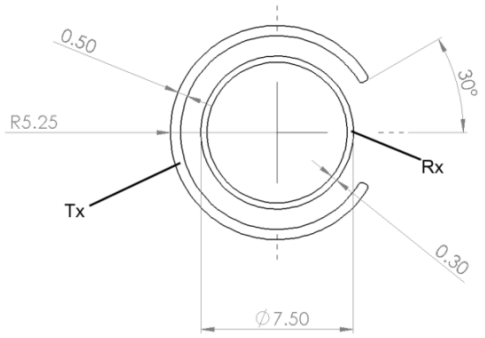
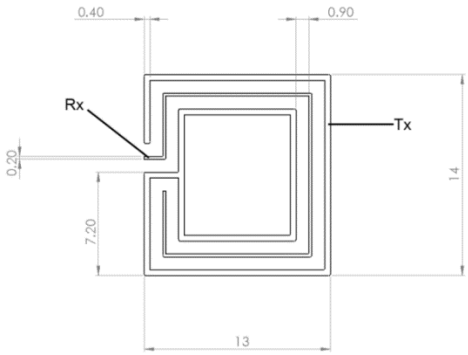
SUM: es la suma de todos los términos.

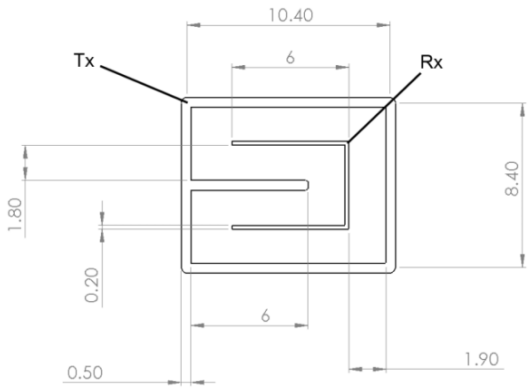
5.2.1. Diseño de botones táctiles

Cada fabricante de tecnología de sensado capacitivo ha caracterizado distintos diseños de sensores de acuerdo a la manera en la que desarrollado sus dispositivos.

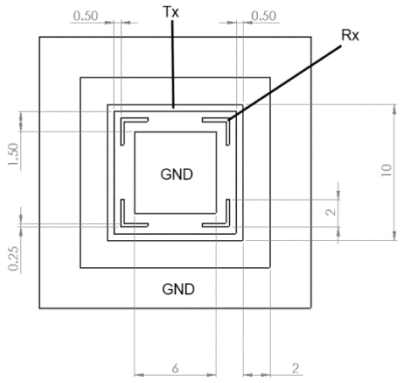
En la bibliografía de Azoteq (Azoteq, 2014) podemos encontrar distintos ejemplos de diseño, de mutual capacitance, como los que se muestran en la Tabla 2. Donde todas las dimensiones están en milímetros, Tx es el electrodo transmisor y Rx el receptor.

Tabla 2: Ejemplos de diseños caracterizados [tomado de (Azoteq, 2014)*]*

 Diseño 1	 Diseño 2
 Diseño 3	 Diseño 4



Diseño 5



Diseño 6

Los resultados de la caracterización podemos observarlos en la Tabla 3. Donde un * implica una baja calificación y *** es la calificación más alta a las características que fueron evaluadas.

Tabla 3: Caracterización de diseños de sensores [tomado de (Azoteq, 2014)*]*

Carac. / Diseño	**Proximidad**	**Touch**	**Panel 2mm**	**Panel 5mm**	**LEDs**	**Detección de agua**	**Inmunidad al agua**
1	***	**	***	***	***	**	*
2	**	***	***	**	***	**	**
3	*	**	**	*	***	*	**
4	**	**	**	*	***	*	***
5	*	**	**	**	***	*	***
6	*	***	***	**	***	*	***

5.2.2. Diseño de trackpads

Actualmente se han desarrollado diferentes maneras en la que los sensores *mutual* pueden ser organizados. Esto involucra la forma, tamaño, separación y distribución en la que son colocados para lograr implementar una superficie táctil. Esta manera de organizarlos generalmente podemos encontrarlo con el término de “patrón”. Uno de los patrones más comunes es el de diamantes (Figura 17).

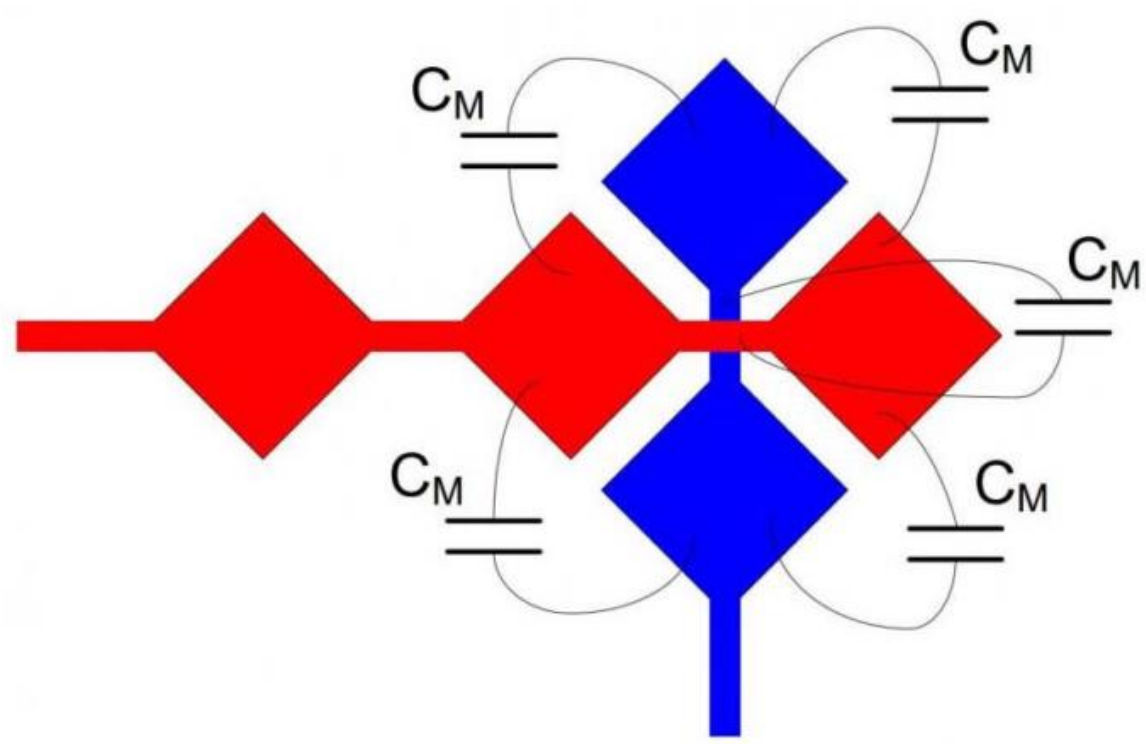


Figura 17: Patrón de diamantes [tomada de (Walker, 2014)*]*

Para lograr tener un buen diseño, la mayoría de los fabricantes provee la documentación necesaria. Una de las metodologías que podemos encontrar es la de Azoteq®. La Figura 18 muestra el flujo de diseño que debe seguirse para lograr un buen diseño (Azoteq, 2013).

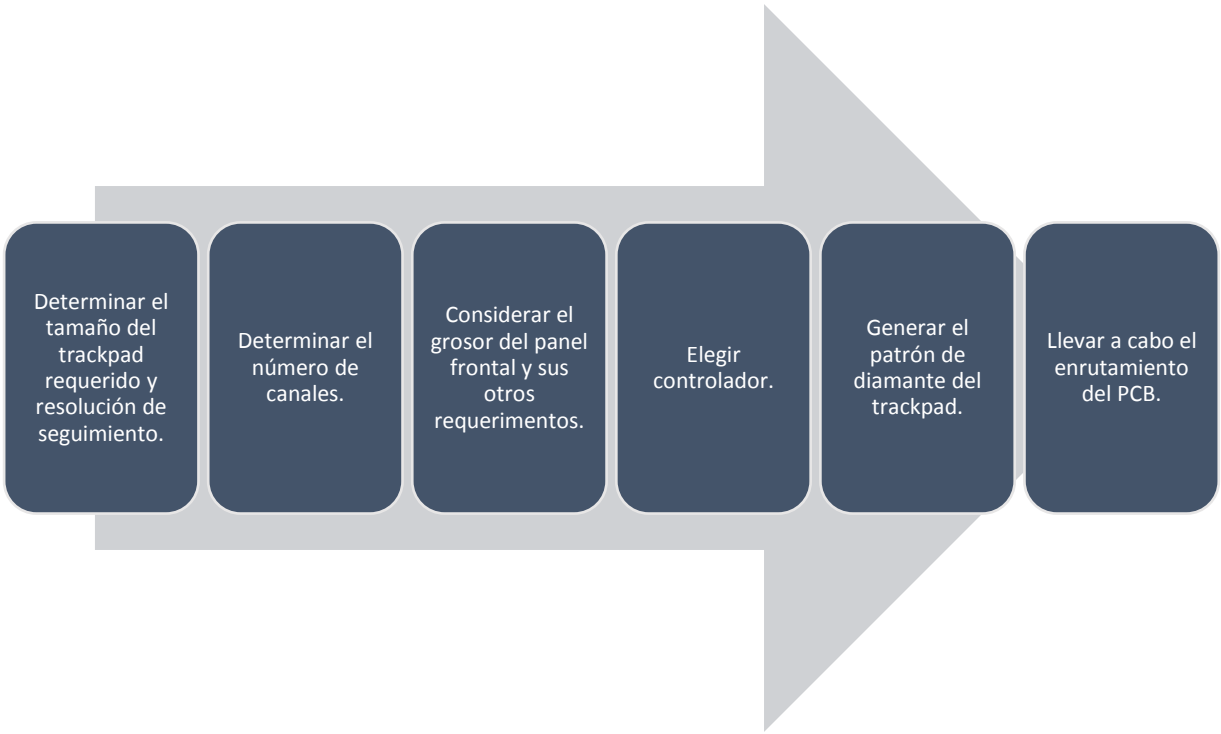


Figura 18: Flujo de diseño de trackpads

Podemos enlistar las elecciones clave de diseño que debemos considerar cuando diseñamos un trackpad:

- Tamaño.
- Rendimiento: Resolución + distancia mínima de “pellizco” (gesto de las manos)
- Estructura y composición del panel frontal.
- Diseño PCB.
- Carcasa mecánica (no será abarcado en este documento).

Para poder describir los criterios de diseño es necesario entender la terminología utilizada que se representa en la Figura 19.

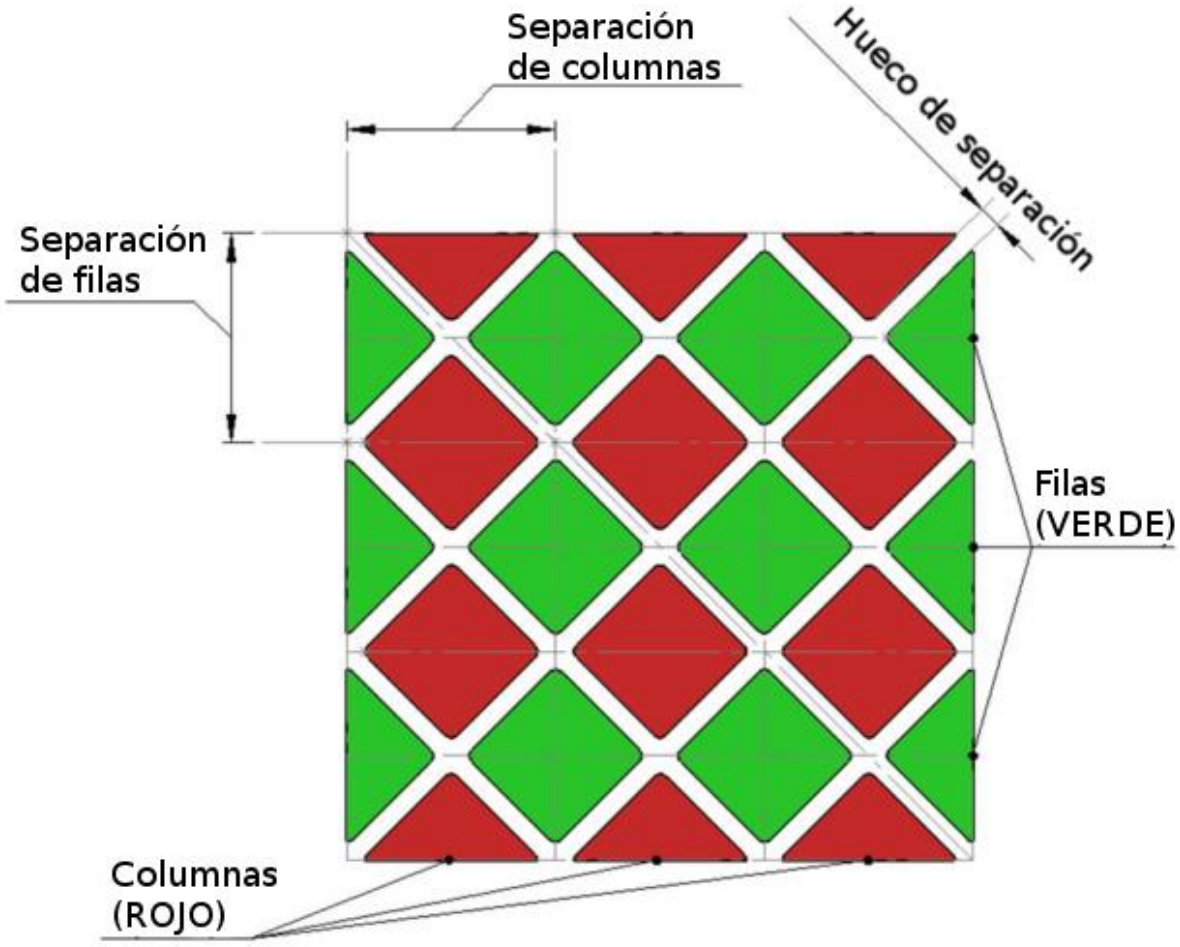


Figura 19: Terminología para el diseño del patrón de diamantes [tomada de (Azoteq, 2013)*].*

Cuando tenemos un número fijo de canales ya establecidos y el tamaño del trackpad es una limitante, la separación entre diamantes debe ser una variable ajustada para encontrar el valor adecuado para las dimensiones del trackpad deseado. El tamaño de la separación de los diamantes también está limitado por el grosor del panel frontal, por lo que debe ser menor al grosor del panel.

Para paneles frontales de hasta 2 mm de grosor, el hueco de separación debe ser aproximadamente $\frac{2}{3}$ del grosor de la cubierta.

La distancia de “pellizco” está definida como la mínima distancia que permite la correcta detección de múltiples toques en estrecha proximidad de uno a otro. Es

deseable que esta distancia sea menor que la distancia de dos dedos juntos. Si la distancia de pellizco deseada no es la adecuada, el número de canales necesita ser incrementado con respecto a esta distancia, mientras se decrementa la separación entre diamantes. La distancia pellizco mínima es 2.5 veces la separación de diamantes.

El panel frontal es uno de los componentes más importantes para el correcto funcionamiento del diseño del trackpad. Los huecos de aire deben ser removidos completamente si es posible. Si el menor hueco de aire está presente, el estrés mecánico no debe causar que esos espacios cambien su tamaño. El panel frontal no debe ser conductivo. En el caso de que se utilicen múltiples materiales, deben estar firmemente conectados.

Cuando se termina el diseño del trackpad, continúa otra parte importante que es el diseño PCB, es importante considerar lo siguiente:

- Es recomendado utilizar PCB FR4 de dos capas con un grosor 1.6 mm.
- Implementar un 40%, 45 grados de tierra en forma de malla en la parte trasera del PCB, esto con el fin de evitar que la mano del usuario sea detectada por la parte de atrás del trackpad (Figura 20).
- Es recomendable el uso de máscara antisoldante.
- Minimizar los cruces entre pistas de TX y RX, si no se puede evitar, deben mantenerse en 90 grados y mantenerlos lo más lejos posible (Figura 21).
- Evitar tener una tierra sólida detrás del patrón de diamantes, ya que puede reducir significativamente la sensibilidad.
- No se debe permitir que las pistas de TX y RX estén en paralelo a menos que una pista de tierra separe los dos grupos, de otra manera el acoplamiento entre TX y RX incrementará la capacitancia parásita en los electrodos y disminuirá la sensibilidad (Figura 21).

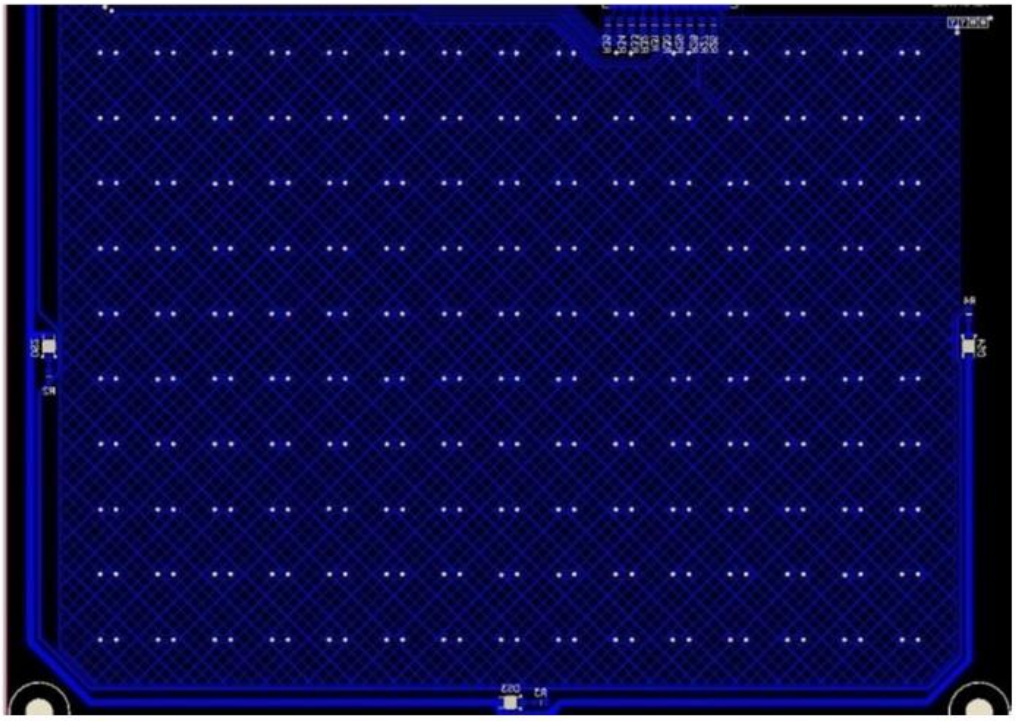


Figura 20: Tierra en forma de malla [tomada de (Azoteq, 2013)*]*

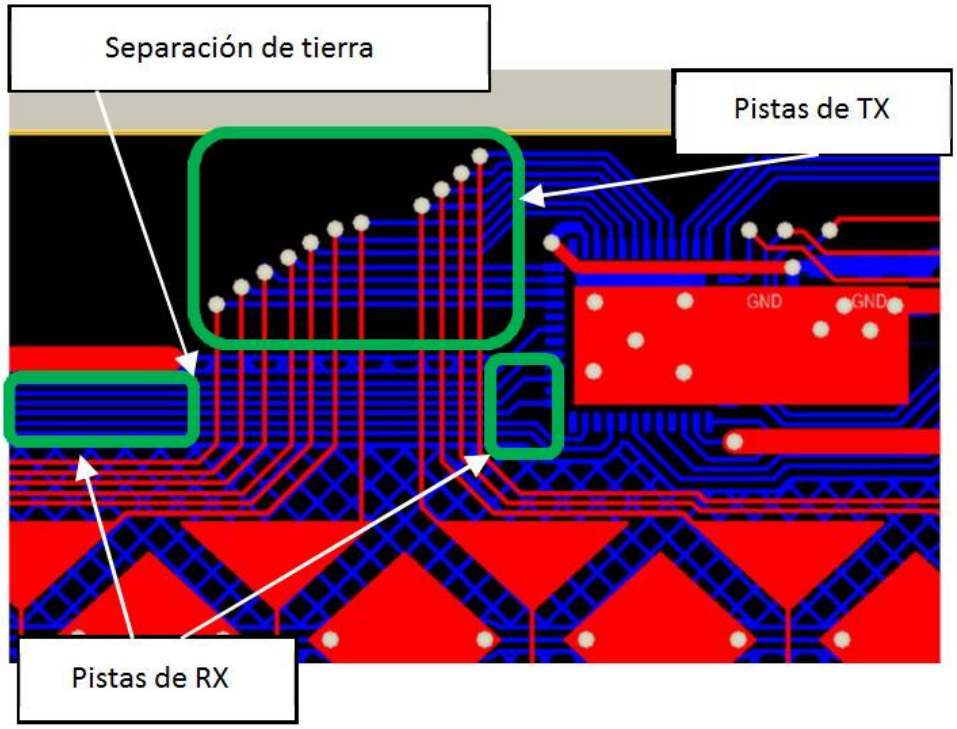


Figura 21: Recomendaciones de diseño PCB para trackpads [tomada de (Azoteq, 2013)*]*

5.3. Protocolo I²C

El bus I²C es un protocolo de comunicación diseñado para la comunicación entre circuitos integrados desarrollado por Philips ® (Philips Semiconductors, 2000). Dos líneas de comunicación, serial data (SDA) y serial clock (SCL), llevan información entre los dispositivos conectados al bus. Cada dispositivo es reconocido por una única dirección y puede operar como un transmisor o receptor, dependiendo de la función de los dispositivos. También pueden ser considerados como maestros o esclavos cuando realizan transferencia de datos. Un maestro es el dispositivo que inicia una transferencia de datos en el bus y genera las señales de reloj para permitir esa transferencia. Cualquier dispositivo con una dirección se considera un esclavo.

El bus I²C es un bus multi-maestro. Esto significa que más de un dispositivo que sea capaz de controlar el bus puede iniciar una transferencia. Como los maestros son usualmente microcontroladores, consideremos el caso de una transferencia de datos entre dos microcontroladores conectados en el I²C bus (Figura 22).

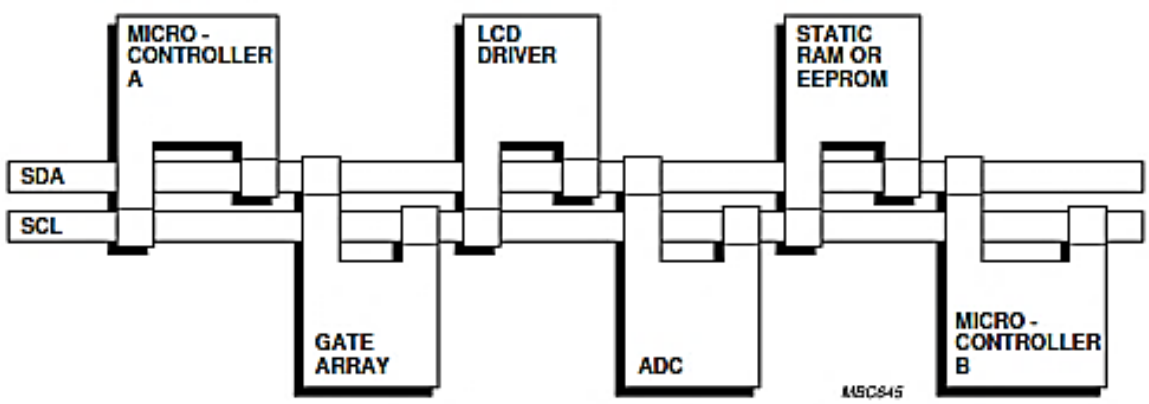


Figura 22: Ejemplo de una configuración del bus I²C usando dos microcontroladores [tomada de (Philips Semiconductors, 2000)*]*

Los puntos importantes que podemos encontrar en el bus I²C está en la relación maestro-esclavo y receptor-transmisor. Se debe notar que esa relación no es permanente, porque solo depende en la dirección del dato transferido en ese momento. La transferencia de datos procede como sigue:

1) Suponiendo que el microcontrolador A quiere mandar información al microcontrolador B:

    - El microcontrolador A (maestro) direcciona al microcontrolador B (esclavo).
    - El microcontrolador A (maestro-transmisor) envía el dato al microcontrolador B (esclavo-receptor).
    - El microcontrolador A termina la transferencia.

2) Si el microcontrolador A quiere recibir información del microcontrolador B:

    - El microcontrolador A (maestro) direcciona al microcontrolador B (esclavo).
    - El microcontrolador B (esclavo-transmisor) envía el dato al microcontrolador A (maestro-receptor).
    - Microcontrolador A termina la transferencia

Incluso en este caso, el maestro (microcontrolador A) genera el tiempo y el fin de la transferencia. Siempre es responsabilidad del maestro la generación de la señal de reloj.

5.3.1. Características generales

Tanto SDA y SCL son líneas bidireccionales conectadas a una fuente positiva de voltaje a través de una fuente de corriente o resistencia “pull-up” (Figura 23). Cuando el bus está libre, ambas líneas están en ALTO. La salida de los estados de los dispositivos conectados al bus debe tener un drenaje o colector abierto para que efectuar la función “wired-AND”. Los datos en el bus I^2C pueden ser transferidos a tasas de hasta 100 kbit/s en el modo “Standard”, hasta 400 kbit/s en el modo “Fast”, o

hasta 3.4 Mbit/s en el modo “High-speed”. El número de interfaces conectadas al bus es únicamente dependiente en la capacitancia límite del bus de 400pF.

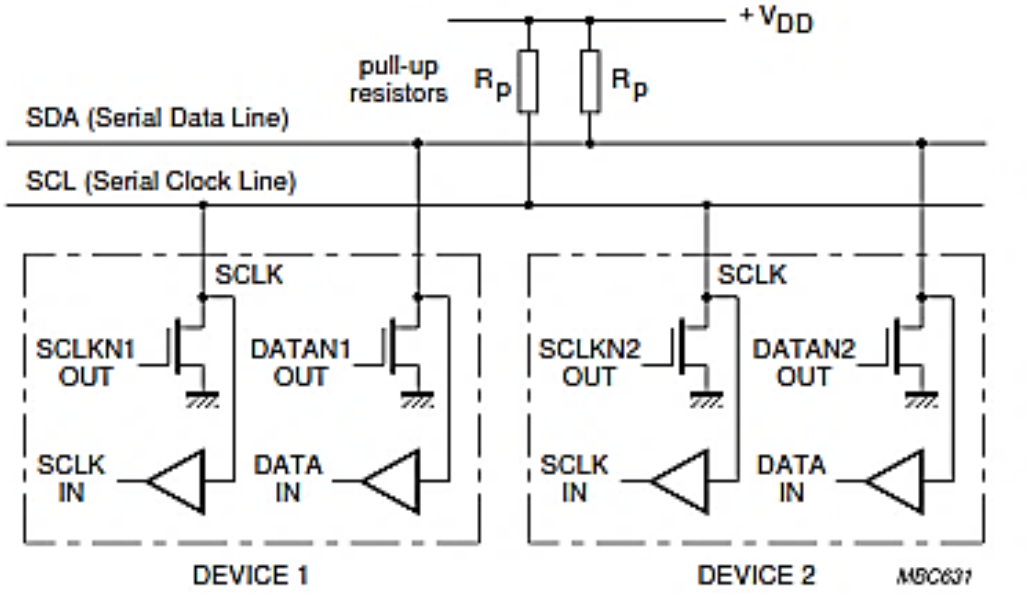


Figura 23: Conexión de dispositivos modo "Standard" y "Fast" al bus I²C [tomada de (Philips Semiconductors, 2000)*]*

5.3.2. Transferencia de bit

Debido a la variedad de dispositivos de diferentes tecnologías (CMOS, NMOS, bipolar) que pueden ser conectados en el bus I²C, los niveles de la lógica ‘0’ (BAJO) y ‘1’ (ALTO) no están ajustados y dependen del nivel asociado a V_{DD}.

Un pulso de reloj es generado por cada bit de dato transferido. El estado ALTO o BAJO de SDA solo puede cambiar cuando la señal de reloj en la línea SCL está en BAJO (Figura 24).

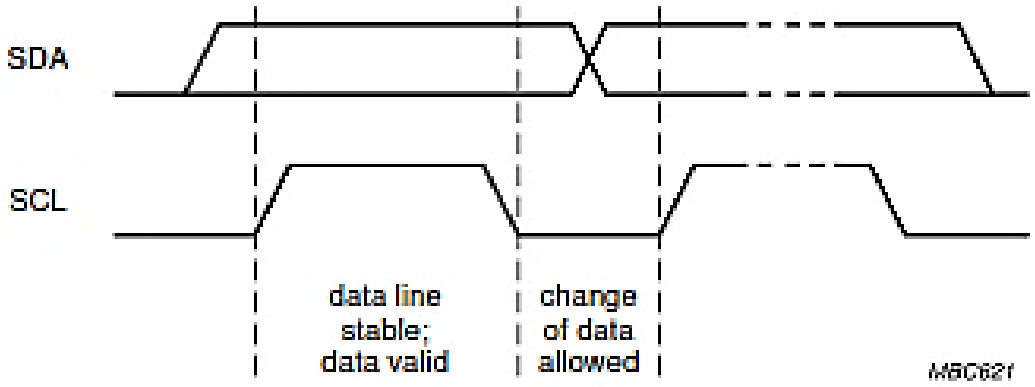


Figura 24: Transferencia de bit en el bus I²C [tomada de (Philips Semiconductors, 2000)*]*

En el procedimiento del bus I²C, se presentan situaciones únicas que están definidas como condiciones de START (S) y STOP (P) (Figura 25).

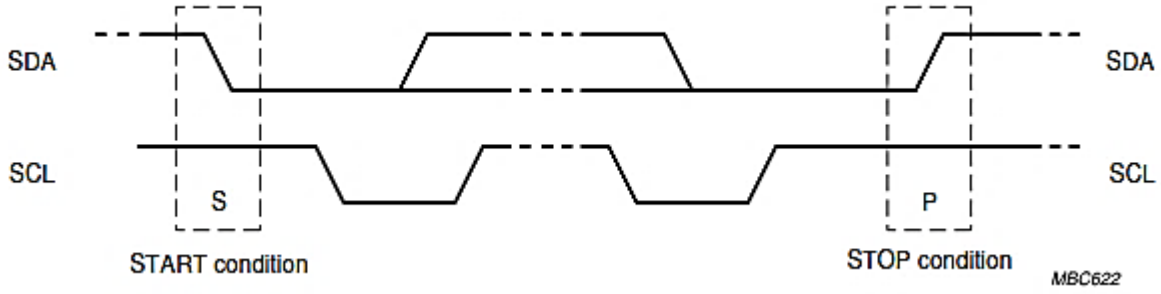


Figura 25: Condiciones de START y STOP [tomada de (Philips Semiconductors, 2000)*]*

Una transición de ALTO a BAJO en la línea de SDA mientras SCL es ALTO indica una condición de START.

Una transición de BAJO a ALTO en la línea de SDA mientras SCL es ALTO define una condición de STOP.

Las condiciones de START y STOP son siempre generadas por el maestro. El bus es considerado ocupado después de la condición de START. El bus es considerado libre de nuevo un cierto tiempo después de la condición de STOP.

El bus permanece ocupado si una repetición de START (Sr) es generada en lugar de una condición de STOP.

5.3.3. Transfiriendo datos

Todo byte puesto en la línea de SDA debe ser de 8 bits. El número de bytes que pueden ser transmitidos por transferencia no está restringido. Cada byte es seguido por un bit de reconocimiento (acknowledge en inglés). Usualmente el dato es transferido con el bit más significativo (MSB) primero (Figura 26). Si un esclavo no puede recibir o transmitir algún otro byte completo de datos hasta que no se haya realizado alguna otra función, por ejemplo, atendiendo una interrupción interna, puede mantener la línea de reloj SCL en BAJO para forzar al maestro entrar en un estado de espera. La transferencia de datos continúa cuando el esclavo está listo para otro byte de datos y libera la línea de reloj SCL.

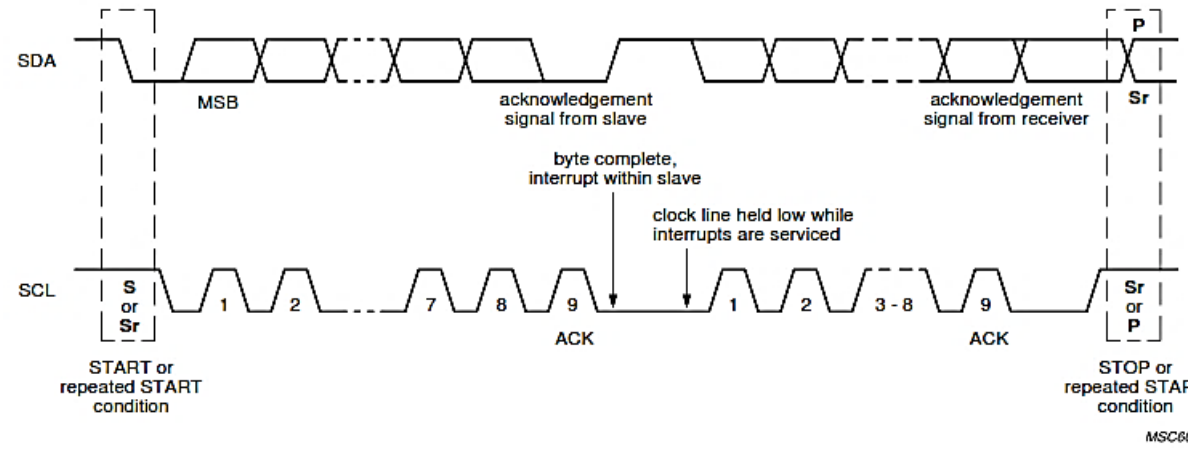


Figura 26: Transferencia de datos en el bus I²C [tomada de (Philips Semiconductors, 2000)*]*

La transferencia de datos con acknowledge es obligatorio. El pulso de reloj relacionado con el acknowledge es generado por el maestro. El transmisor libera la línea SDA (ALTO) durante el pulso de reloj del acknowledge.

El receptor debe poner hacia abajo la línea de SDA durante el pulso de reloj del acknowledge hasta que permanezca establemente en BAJO durante el periodo en ALTO de este pulso de reloj (Figura 27).

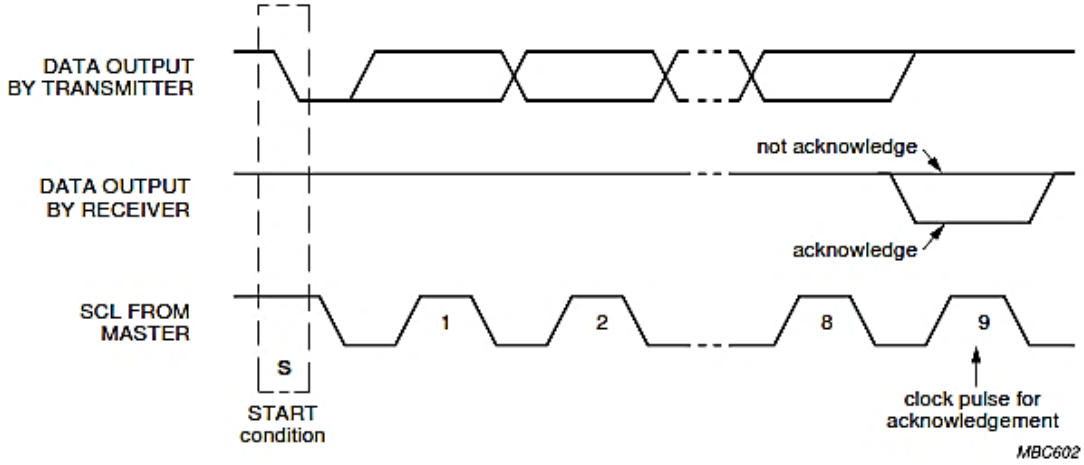


Figura 27: Acknowledge en el bus I²C [tomada de (Philips Semiconductors, 2000)*]*

Usualmente, cuando un receptor ha sido direccionado está obligado a generar un acknowledge después de cada byte que recibe.

Cuando un esclavo no responde con un acknowledge a la dirección de esclavo (por ejemplo, es incapaz de recibir o transmitir porque está realizando alguna función en tiempo real), la línea de datos se debe dejar en ALTO por el esclavo. El maestro puede generar en dado caso una condición de STOP para abortar la transferencia, o una repetición de la condición de START para iniciar una nueva transferencia.

Si un esclavo-receptor devuelve el acknowledge a la dirección de esclavo, pero, algún tiempo después en la transferencia no puede recibir algún byte de datos más, el maestro debe abortar de nuevo la transferencia. Esto es indicado por el esclavo generando el not-acknowledge en el primer byte siguiente. El esclavo deja la línea de datos en ALTO y el maestro genera una condición de STOP o repetición de START.

Si un maestro-receptor está envuelto en una transferencia, debe señalar el fin de dato al esclavo-transmisor evitando la generación de un acknowledge en el último byte que fue recibido del esclavo. El esclavo transmisor debe liberar la línea de datos para permitir al maestro que genere una condición de STOP o repetición de START.

5.3.4. Formatos con direcciones de 7 bits

La transferencia de datos sigue el formato mostrado en la Figura 28. Después de la condición de START (S), una dirección de esclavo es enviada. Esta dirección es de 7 bits de largo seguido por un octavo bit que es el bit de dirección de dato $(R/\overline{W})$, un 'cero' indica una transmisión (WRITE) y un 'uno' indica una solicitud para leer datos (READ). Una transferencia de datos siempre es terminada por una condición de STOP (P) generada por el maestro. Sin embargo, si un maestro requiere mantener la comunicación en el bus, puede generar una repetición de la condición de START (Sr) y direccionar otro esclavo sin generar una condición de STOP primero. Varias combinaciones de formatos de escritura/lectura son entonces posibles dentro de dicha transferencia.

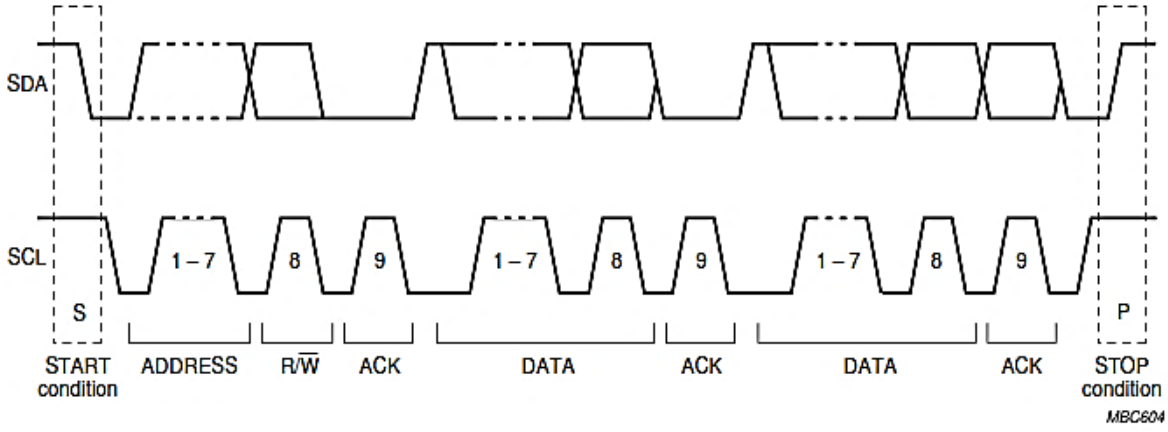


Figura 28: Una transferencia de datos completa por el bus I²C [tomada de (Philips Semiconductors, 2000)*]*

Los posibles formatos de transferencia de datos son:

- El maestro-transmisor transmite al esclavo-receptor. La dirección de transferencia no cambia (Figura 29)

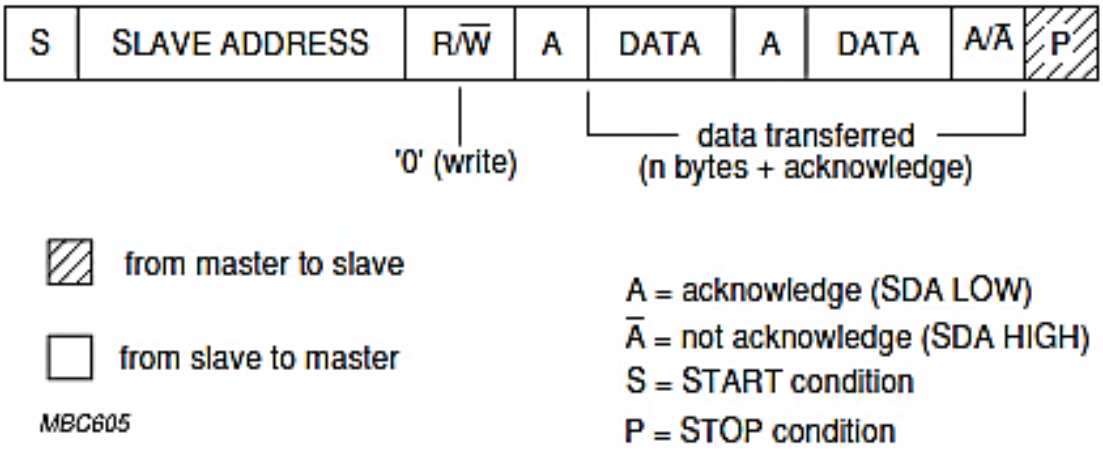


Figura 29: Un maestro-transmisor direccionando un esclavo receptor con una dirección de 7 bits [tomada de (Philips Semiconductors, 2000)*].*

- El maestro lee el esclavo inmediatamente después del primer byte (Figura 30). Al momento del primer acknowledge, el maestro-transmisor se convierte en un maestro-receptor y el esclavo-receptor se convierte en un esclavo-transmisor. Este primer acknowledge es todavía generado por el esclavo. La condición de STOP es generada por el maestro, que previamente a enviado un not-acknowledge $(\bar{A})$.

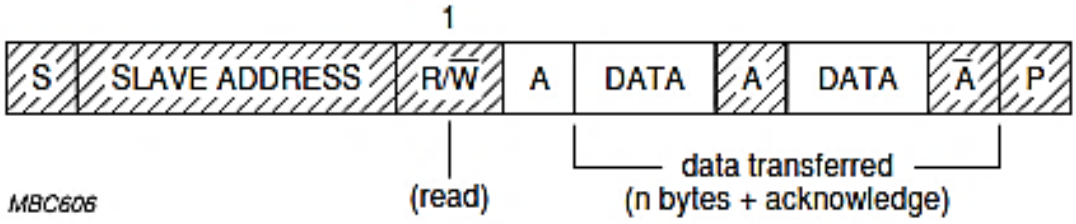


Figura 30: Un maestro lee un esclavo inmediatamente después del primer byte [tomada de (Philips Semiconductors, 2000)*]*

- Formato combinado (Figura 31). Durante un cambio de dirección con una transferencia, la condición de START y la dirección de esclavo son ambos repetidos, pero con el $R/\overline{W}$ bit invertido. Si el maestro receptor envía una repetición de la condición de START, entonces ha enviado previamente un not-acknowledge $(\bar{A})$.

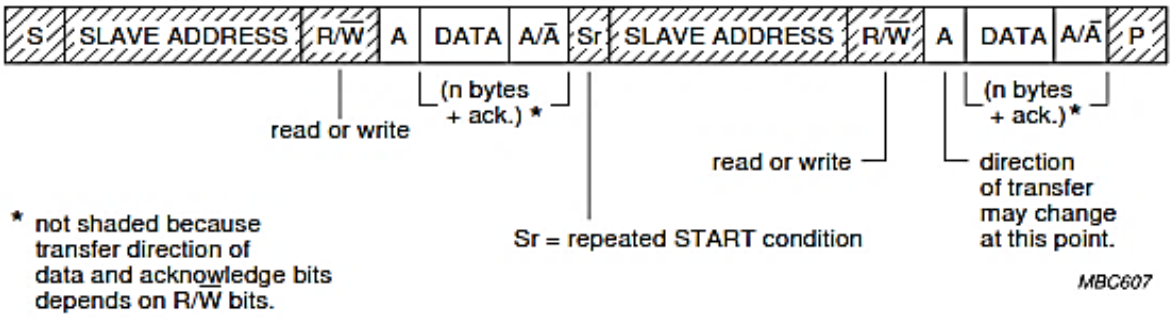


Figura 31: Formato combinado de transferencia de datos por I²C [tomada de (Philips Semiconductors, 2000)*]*

5.4. Puerto serial UART

El UART (*Universal Asynchronous Receiver/Transmitter*) es un protocolo de comunicación que transmite y/o recibe datos de manera serial asíncronamente, es decir, que cada paquete datos es transmitido por un solo hilo o nodo electrónico sin necesidad de una señal de reloj (Axelson, 2007).

Cada paquete de datos en cada transacción está conformado por un bit Start, bits de la información útil (dato o *payload*), un bit de paridad opcional, y uno o más bits Stop. Por ejemplo, en la Figura 32 se transmite el dato 61H (hexadecimal), en el que inicialmente se manda el bit de Start, la secuencia de bits del payload iniciando por el bit menos significativo (LSb), y un bit de Stop. Esta configuración de transferencia es muy común, por cada paquete de datos se transfieren 8 bits de información, no se usa el bit de paridad y se manda un solo bit de Stop.

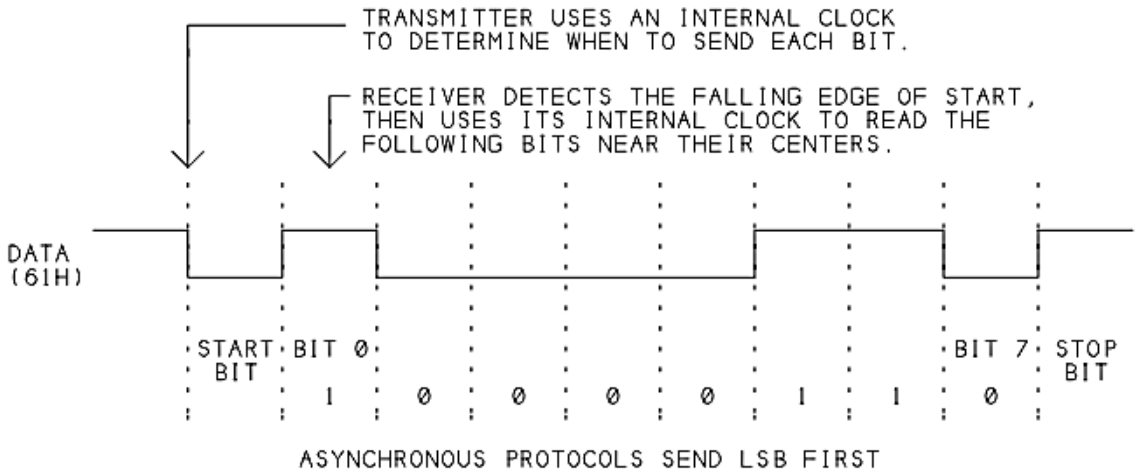


Figura 32: Las transmisiones asíncronas requieren que cada dispositivo tenga su propio reloj para enviar adecuadamente la información [tomada de (Axelson, 2007)*]*

El bit de paridad puede proporcionar una forma básica de detectar errores. Con paridad par, los bits de datos y el bit de paridad en cada palabra contienen un número par de 1's. Con paridad impar, los bits de datos y el bit de paridad en cada palabra contienen un número impar de 1's.

La velocidad de transferencia puede ser medido en bits por segundos transmitidos o recibidos por unidad de tiempo, usualmente expresado en bits por segundo (bps). Aunque es más común usar la velocidad de baudios, que es el número de posibles eventos, o transiciones de datos, por segundo. El término de velocidad de baudios usualmente se refiere a la velocidad de bits. Por ejemplo, a 9600 bps el tiempo que dura un bit es de 104 µs.

Un puerto que utiliza el protocolo UART debe contar con al menos dos señales de comunicación: TX y RX. TX indica la línea en el que un dispositivo envía datos y RX la línea en el que se reciben datos desde otro dispositivo.

Actualmente varios microcontroladores cuentan con el protocolo embebido en su hardware interno. En el que, para enviar un byte, se escribe el byte a transmitir en el buffer de transmisión del puerto seleccionado. El UART entonces envía el dato, bit por bit en el formato solicitado, agregando Stop, Start, y bit de paridad si es necesario.

De manera similar, el UART guarda los bytes recibidos en un buffer. Después de recibir un byte, el UART puede generar una interrupción al dispositivo de la recepción del dato, o el software puede leer el puerto para verificar si se recibió un dato (polling).

Cuando el puerto del UART está libre, la salida del transmisor es un 1 lógico. Para indicar el inicio de la transmisión, el transmisor envía un 0 lógico de un bit de ancho, esto es el bit Start. Después del bit Start, el transmisor envía los 8 bits del dato en secuencia, comenzando con el LSb. El transmisor envía entonces un 1 lógico, el cual funciona como bit Stop. Inmediatamente seguido del bit Stop o en cualquier momento después, el transmisor puede enviar un nuevo bit Start para iniciar una nueva palabra de transmisión.

Para el receptor, la transición desde 1 lógico al bit de Start de 0 lógico indica que una nueva palabra está llegando. La transición y la velocidad de bits determinan el tiempo para detectar los bits siguientes. Los intentos del receptor para leer el estado lógico de cada bit cerca de la mitad del periodo de tiempo del bit. Leyendo en la mitad del periodo ayuda a asegurar que el receptor detecte el valor del bit correctamente aun si el reloj de transmisión y recepción no coinciden exactamente en frecuencia y fase.

Para reducir errores, muchos UARTs toman tres muestras en medio de cada bit y usan los niveles lógicos tal que coincidan dos o más de las muestras. Para evitar detectar breves interferencias de ruido como bits Start, algunos UARTs leen el bit Start una segunda vez a la mitad del bit y aceptan el bit Start solo si el bit restante es un 0 lógico.

El control de flujo es una forma de asegurar que los datos lleguen sin errores. Con control de flujo, un transmisor puede indicar cuando tiene un dato para enviar, y un receptor puede indicar cuando está listo para recibir datos. Los computadores en un enlace serial deberían usar control de flujo a menos que los buffers de recepción sean los suficientemente largos para mantener todos los datos que puedan llegar antes que la computadora receptora pueda leer los datos desde el búfer.

Una forma común de control de flujo por hardware, es que el receptor configura una línea dedicada a un estado definido cuando está listo para recibir datos. El transmisor checa el estado de la línea antes de enviar el dato. Si la línea no está en estado de

espera, el transmisor envía el dato. El control de flujo en ambas direcciones requiere una línea para cada dirección. Algunas veces este método también es llamado *handshaking*. Sin embargo, un *handshake* completo requiere comunicación de 2 vías: el transmisor indicando que tiene datos para enviar, y el receptor indica cuando está listo para recibir los datos.

La señal entrante es conocida como “Clear To Send” (CTS) y la señal saliente es “Request to Send” (RTS). (Los nombres reflejan un uso alternativo de las líneas para implementar un *handshake* completo.) Un cable que conecta dos PCs debe conectar cada salida RTS a la entrada CTS de la otra computadora. Por ejemplo, un voltaje RS-232 positivo significa listo para recibir y un voltaje negativo significa no listo.

Los microcontroladores típicamente no tienen líneas CTS y RTS dedicadas. El firmware del dispositivo puede usar cualquier pin de puerto disponible para control de flujo.

5.5. Protocolo SPI

La interfaz periférica serial (Serial Peripheral Interface, SPI en inglés) es un protocolo de comunicación desarrollado por Motorola ®. El protocolo SPI se convirtió en un estándar *de facto*, pero no tiene una especificación liberada oficialmente o aceptada por cualquier comité internacional. Eso permite algo de flexibilidad en la implementación del protocolo SPI en los dispositivos electrónicos, pero también requiere flexibilidad programable del microcontrolador anfitrión (Intersil Corporation, 2007).

El protocolo SPI es utilizado para comunicación serial síncrona entre un microcontrolador anfitrión y periféricos. El protocolo requiere dos líneas de control (CS y SCK) y dos líneas de datos (SDI y SDO) como se muestra en la Figura 33.

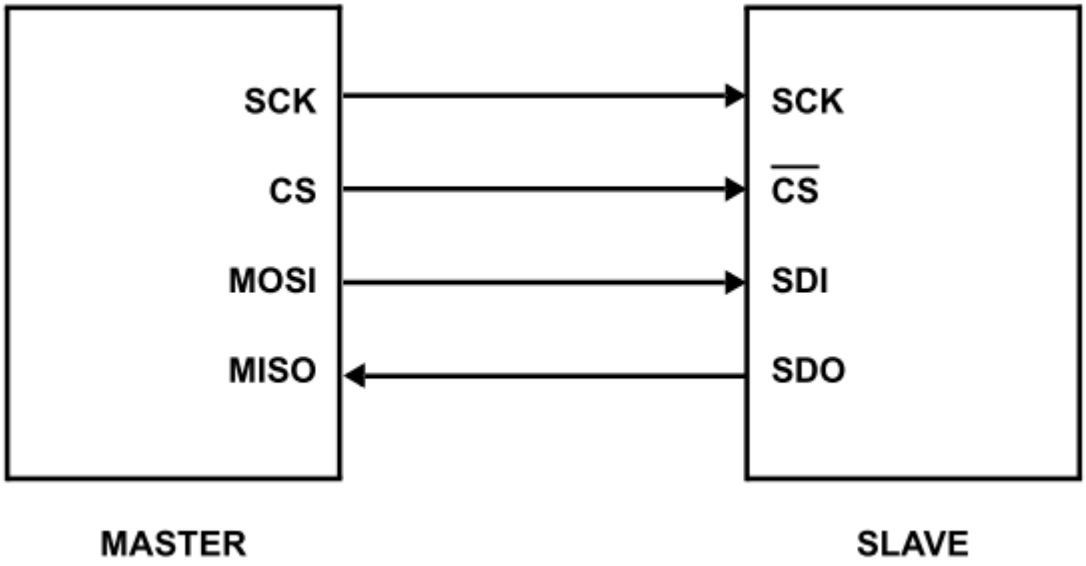


Figura 33: Implementación de un bus SPI maestro-esclavo (master-slave) [tomada de (Intersil Corporation, 2007)*]*

El bus SPI especifica cuatro señales lógicas:

- SCK – *Serial Clock*, proporcionada por el maestro (o *master*).
- CS – *Chip Select*, permite al maestro seleccionar el dispositivo periférico (esclavo o *slave*).
- MOSI/SDI – Master Output Slave Input / Serial Data In, por esta línea el maestro le envía datos al esclavo.
- MISO/SDO – Master Input Slave Output / Serial Data Out, por esta línea el maestro adquiere datos provenientes del esclavo.

Generalmente CS es activo en bajo, es decir, el periférico es seleccionado cuando recibe una señal en bajo o “0” lógico por esta línea. Un maestro, usualmente el microcontrolador anfitrión, siempre provee una señal de reloj para todos los dispositivos en un bus ya sea si está seleccionado o no. Solo un maestro debe estar activo a la vez. El protocolo SPI opera en modo *full-duplex*, cuando se transfieren datos

de entrada y salida en ambas líneas (SDI y SDO) simultáneamente. Los dispositivos que no están seleccionados mantienen sus líneas SDO en alta impedancia e inactivas.

Debido a que el reloj sirve como una sincronización de la comunicación de datos, el protocolo SPI tiene cuatro modos, basados en la polaridad del reloj (CPOL) y en la fase del reloj (CPHA) como se muestra en la Tabla 4 y la Figura 34.

Tabla 4: Modos SPI [tomado de (Intersil Corporation, 2007)*]*

Modo SPI	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Si la fase del reloj es cero (CPHA = 0) el dato es capturado en el flanco de subida del reloj con CPHOL = 0, y en el flanco de bajada del reloj con CPHOL = 1. Si CPHA = 1, las polaridades están invertidas, el dato es capturado en el flanco de bajada del reloj con CPOL = 0, y en el flanco de subida con CPHOL = 1.

Múltiples dispositivos esclavos pueden ser conectados en paralelo utilizando el mismo bus SPI.

Para la conexión en paralelo, cada dispositivo en el bus debe tener una línea CS separada, mientras que las líneas SCK, SDI y SDO están conectadas en paralelo como se muestra en la Figura 35.

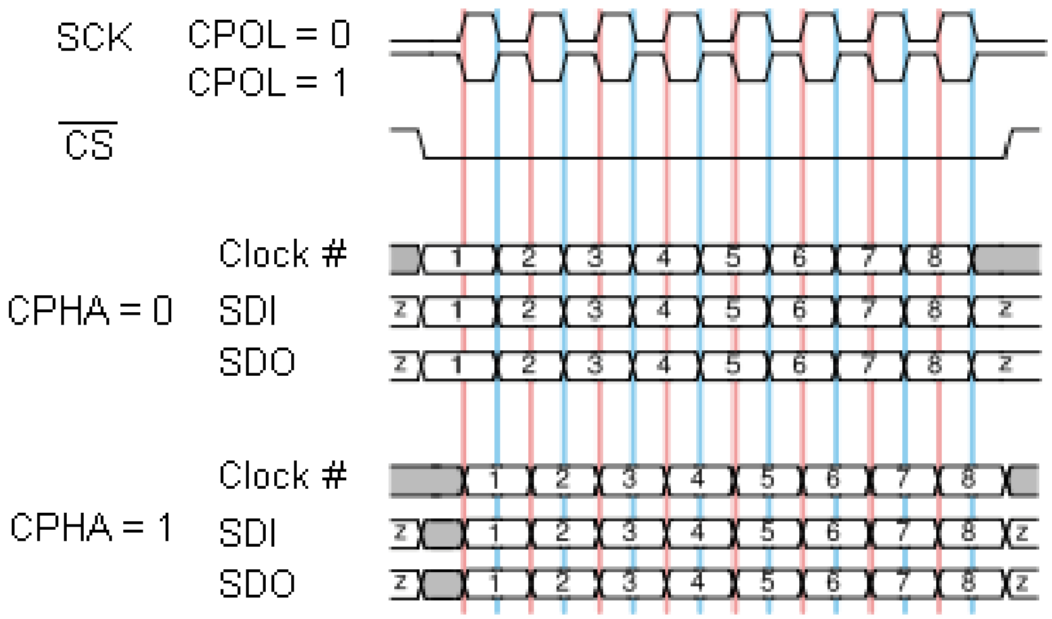


Figura 34:Diagrama de tiempo del protocolo SPI [tomada de (Intersil Corporation, 2007)*]*

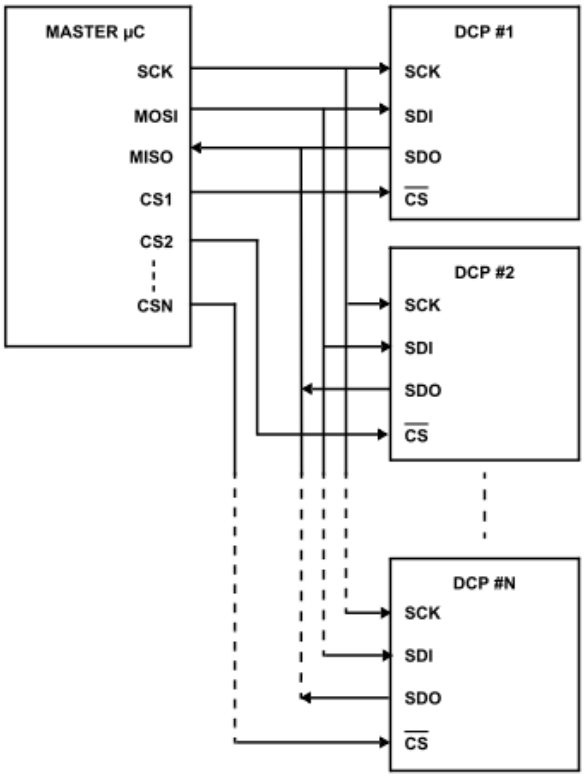


Figura 35: Configuración en paralelo del bus SPI [tomada de (Intersil Corporation, 2007)*]*

5.6. Formato JSON

JSON (JavaScript Object Notation – Notación de objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del lenguaje de programación JavaScript (Standard ECMA-262 3rd Edition – diciembre 1999). Es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, JavaScript, Perl, Python y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos (ECMA, 2013).

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con “{“ (llave de apertura) y termine con “}” (llave de cierre). Cada nombre es seguido por “:” (dos puntos) y los pares nombre/valor están separados por “,” (coma) (Figura 36).

Un arreglo es una colección de valores. Un arreglo comienza con “[“ (corchete izquierdo) y termina con “]” (corchete derecho). Los valores se separan por “,” (coma) (Figura 37).

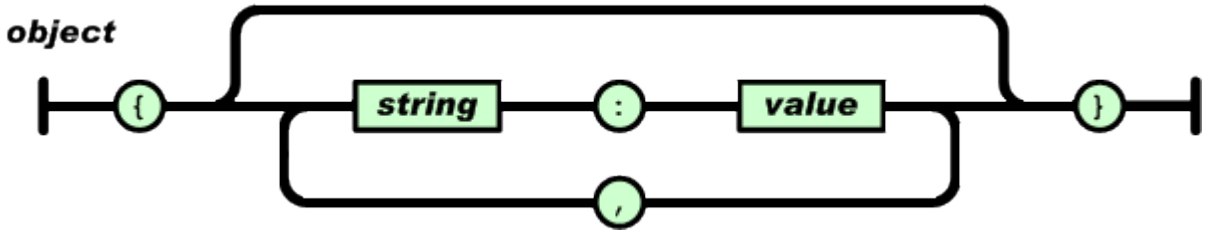


Figura 36: Objeto JSON

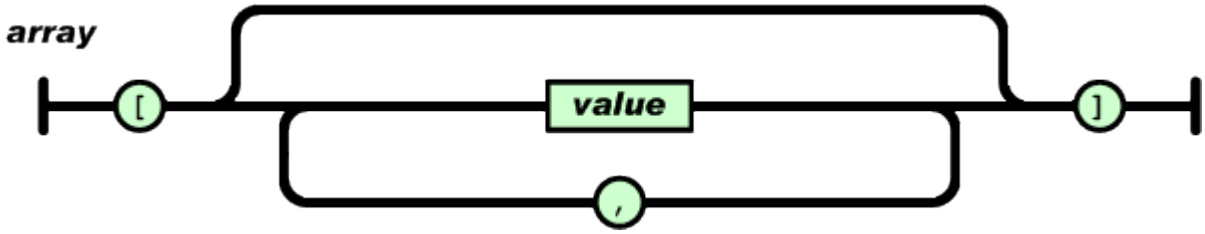


Figura 37: Arreglo JSON [tomada de (ECMA, 2013)*]*

Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse (Figura 38).

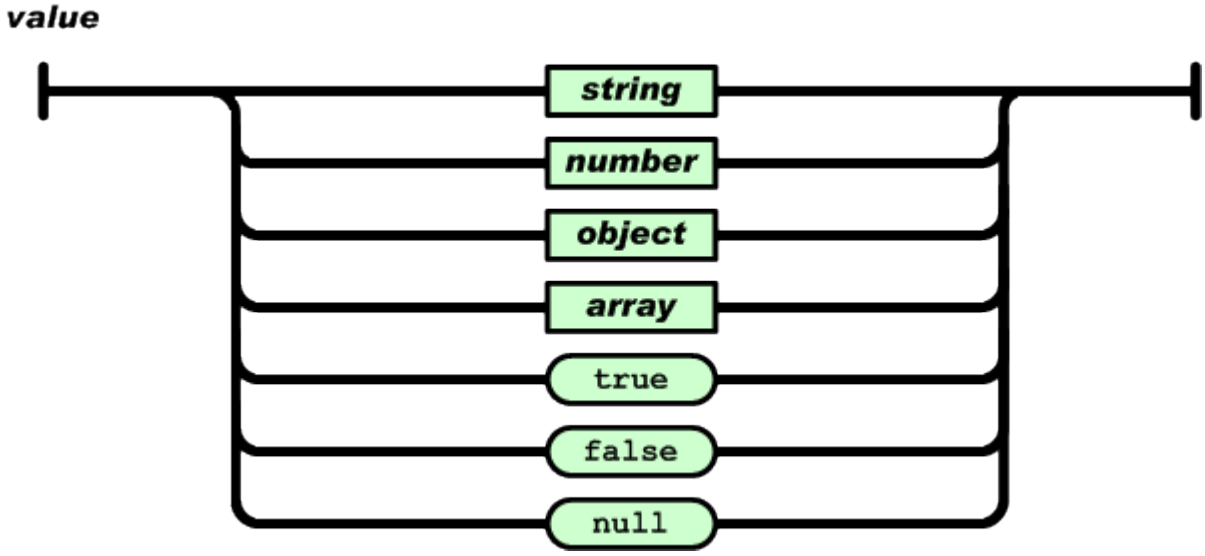


Figura 38: Valor JSON [tomada de (ECMA, 2013)*]*

Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de caracteres es parecida a una cadena de caracteres C o Java (Figura 39).

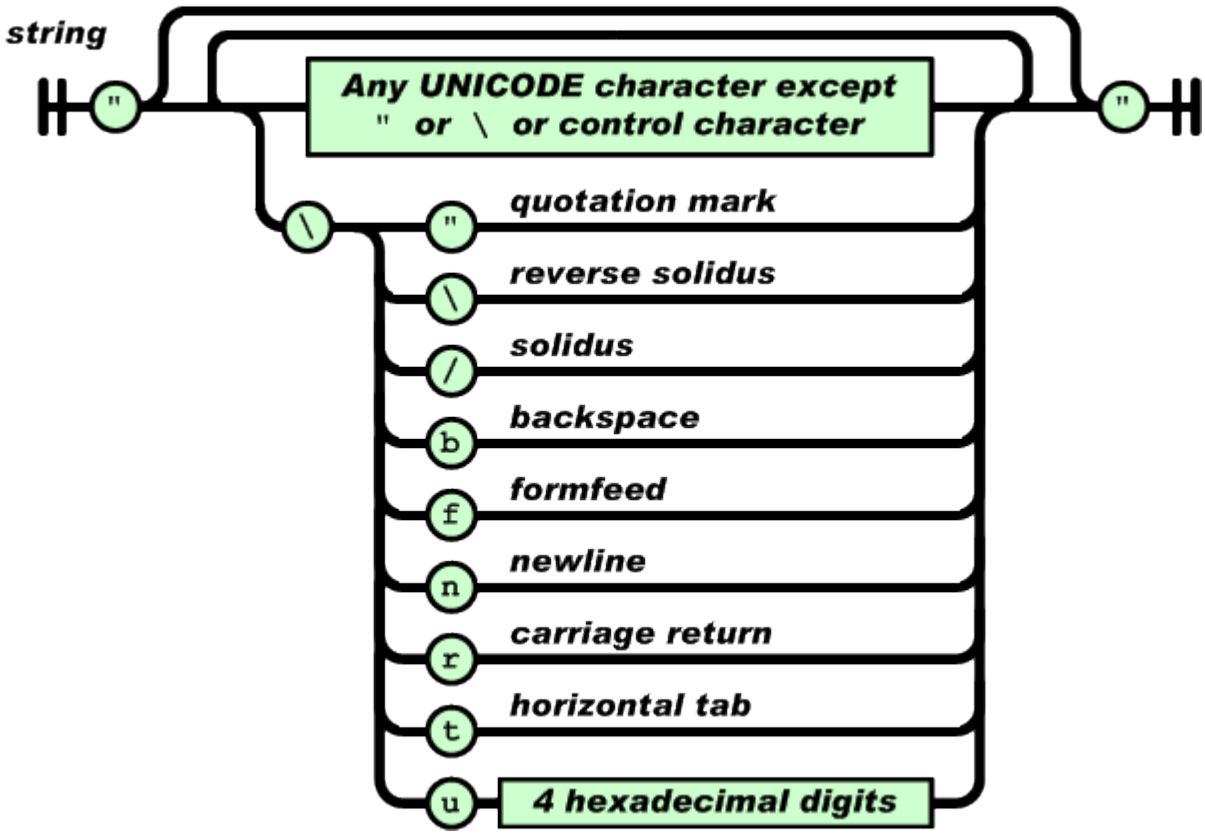


Figura 39: Cadena JSON [tomada de (ECMA, 2013)*]*

Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales (Figura 40).

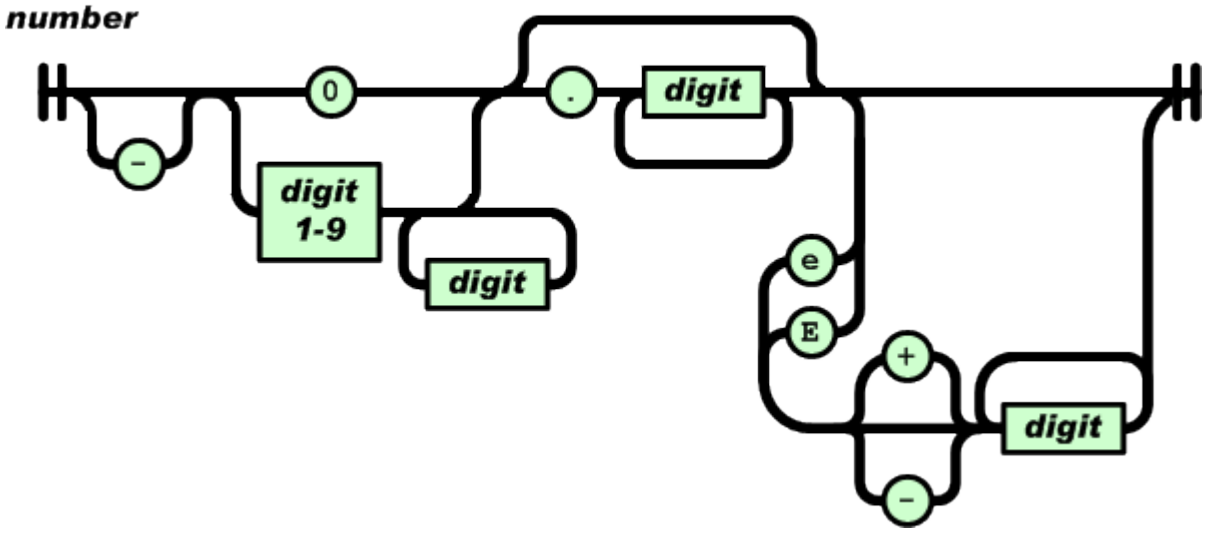


Figura 40: Número JSON [tomada de (ECMA, 2013)*]*

Un ejemplo de un archivo JSON lo podemos observar en la Figura 41.

```
{
    "colorsArray":[{
            "colorName":"red",
            "hexValue":"#f00"
        },
        {
            "colorName":"green",
            "hexValue":"#0f0"
        },
        {
            "colorName":"blue",
            "hexValue":"#00f"
        },
        {
            "colorName":"black",
            "hexValue":"#000"
        }
    ]
}
```

Figura 41: Ejemplo del formato JSON

6. METODOLOGÍA

Para la realización del proyecto se siguió el orden del proceso que se ilustra en la Figura 42.

Primero se llevó a cabo una simulación 3D con el fin de elegir la mejor distribución de los sensores sobre el laberinto. A partir de este resultado, se realizó la selección del controlador que se utilizó para el manejo de los sensores capacitivos. Una vez elegido el controlador de los sensores, se siguieron las recomendaciones de diseño del fabricante de dicho controlador. Para validar el diseño se realizaron pruebas del funcionamiento una vez realizada la manufactura. Posteriormente se realizó el diseño e implementación del sistema electrónico que sirve como interfaz de usuario para configurar experimentos, comunicación con los sensores, fuente de energía y respaldo y transferencia de experimentos.

En los siguientes apartados se explicará detalladamente cada elemento del proceso que se llevó a cabo para realizar el proyecto.

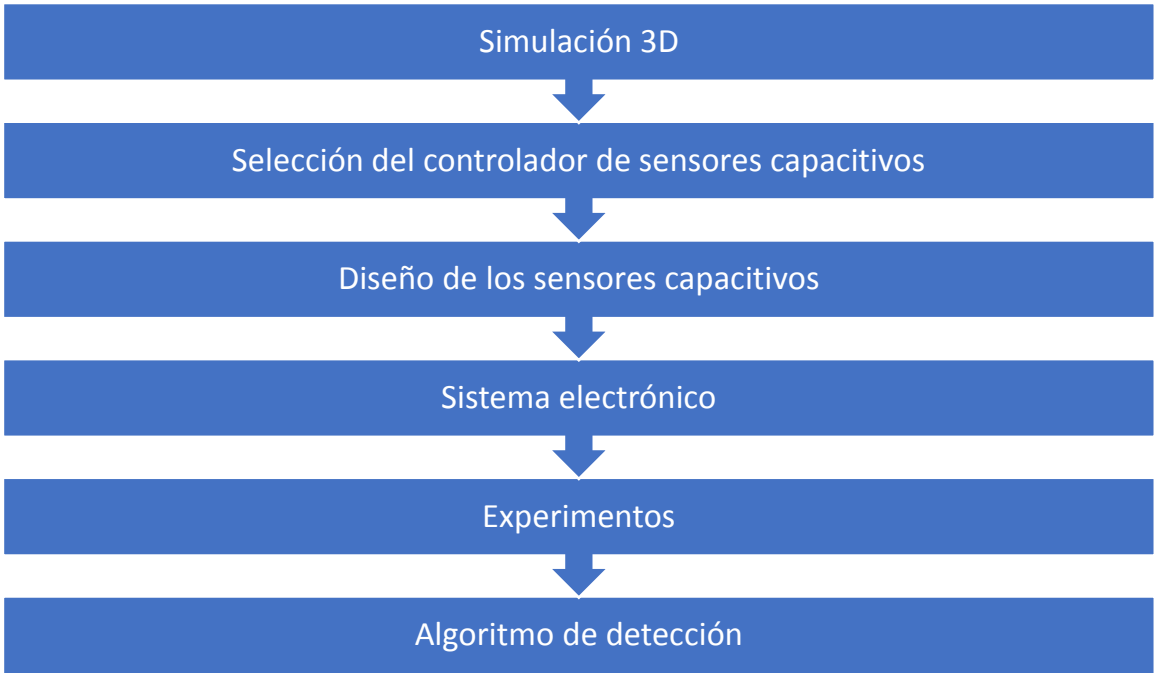


Figura 42: Diagrama cronológico de la metodología

6.1. Simulación 3D

Con el fin de tener una aproximación de la mejor distribución de los sensores para detectar a una rata en la plataforma del laberinto elevado en forma de cruz se realizó una simulación 3D utilizando el software Unity para programación de videojuegos y el software Blender para realizar modelado y animación 3D. Esto es con el fin de identificar con antelación las dimensiones y distribución óptima de los sensores capaces de detectar las poses más comunes que realiza una rata en el laberinto y evitar falsos positivos o falsos negativos al momento de evaluar si el roedor se encuentra dentro o fuera de un brazo, que es una de las desventajas del sistema explicado en el apartado 2.2.1.

6.1.1. Modelo del roedor

Para la simulación se realizó con Blender el modelo 3D del roedor para poder importarlo en la plataforma de Unity. Para poder realizar el modelo de la rata se realizaron mediciones de una rata adulta de tamaño promedio, entre 250 y 300g, utilizada para experimentos relacionados con ansiedad en el laberinto elevado en forma cruz. En la imagen ilustrada en la Figura 43 podemos observar las dimensiones de una rata Wistar macho de 280g.

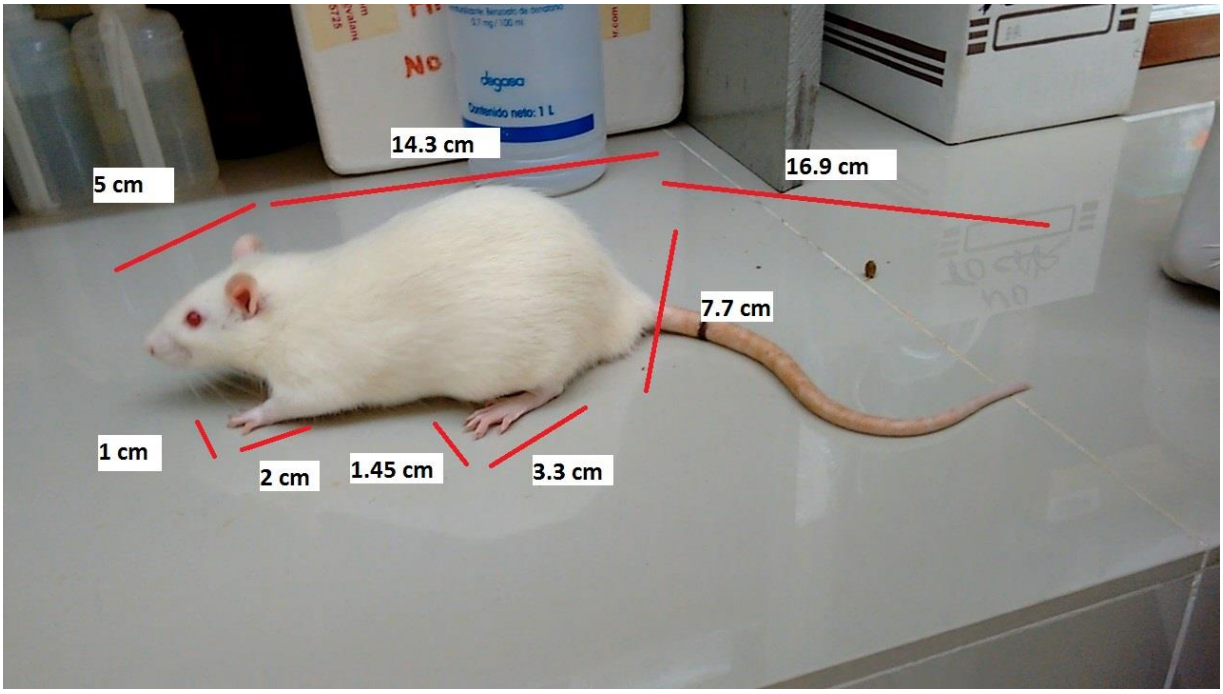


Figura 43: Dimensiones de rata Wistar macho de 280g.

Además, a partir de observación de videos de experimentos, se implementaron las siguientes animaciones al modelo del roedor:

- Caminata con cola abajo (Figura 44).
- Caminata con cola arriba (Figura 45).
- Olfateo (Figura 46).
- Caminata y olfateo (Figura 47).
- Levantamiento en dos patas (Figura 48).
- Giro (Figura 49).

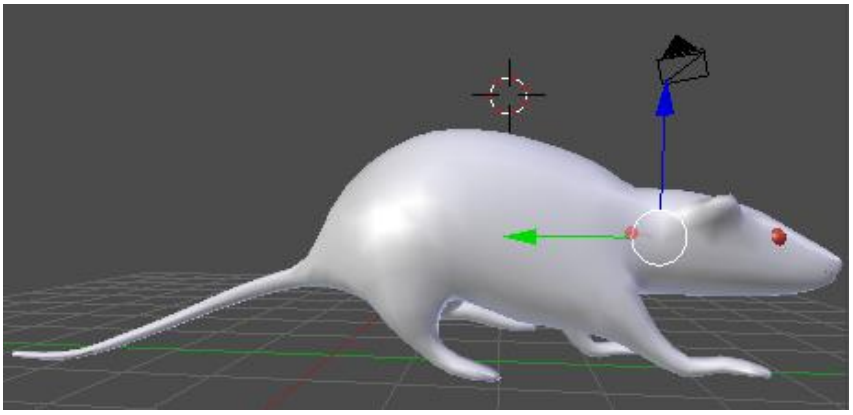


Figura 44: Animación de caminata con cola abajo.

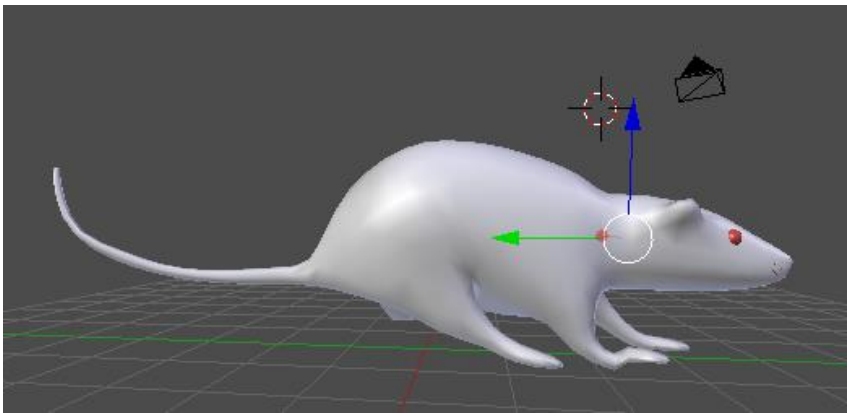


Figura 45: Animación de caminata con cola arriba.

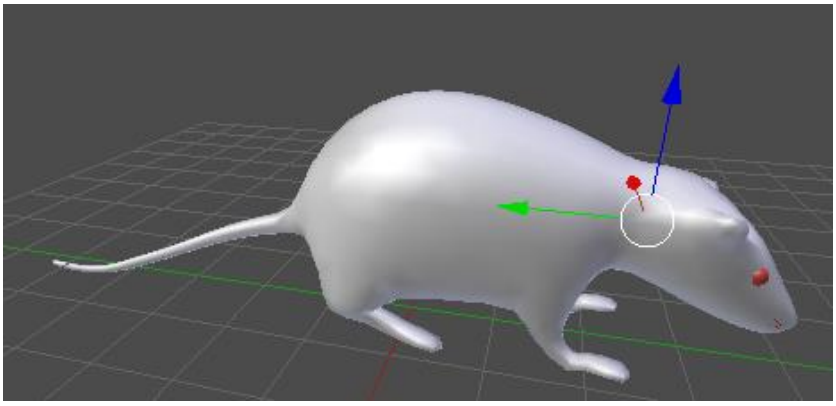


Figura 46: Animación de olfateo.

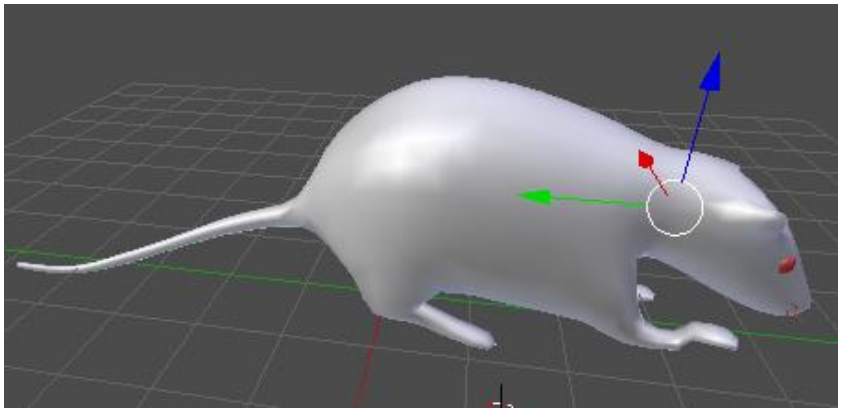


Figura 47: Animación de caminata y olfateo.

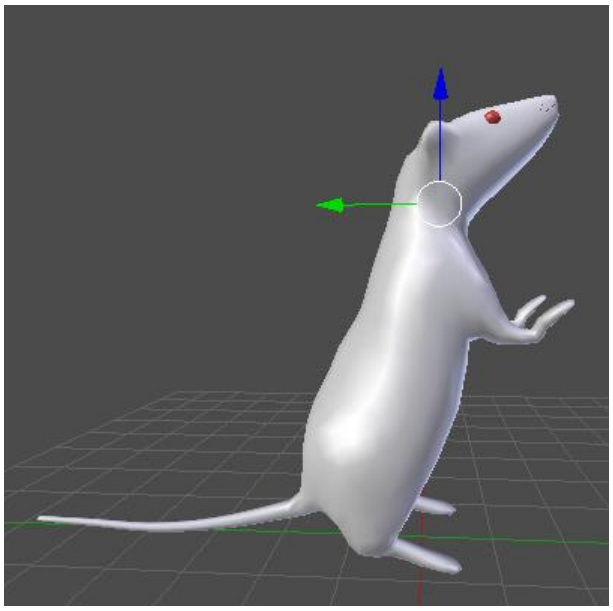


Figura 48: Animación de levantamiento en dos patas.

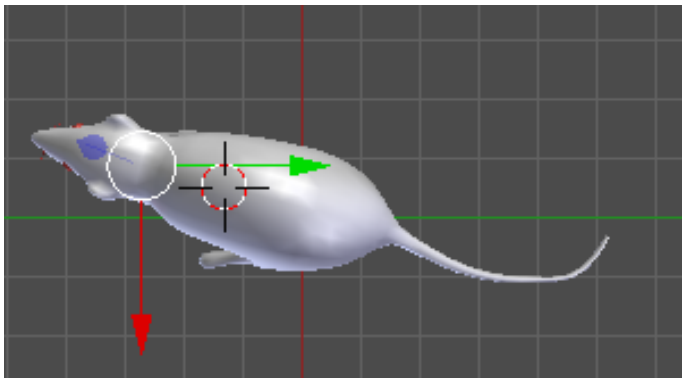


Figura 49: Animación de giro.

6.1.2. Modelo del laberinto y sensores

Después de implementar las animaciones del roedor, el modelo del laberinto y de los sensores se realizaron en Unity (Figura 50), ya que simplemente son figuras geométricas sencillas.

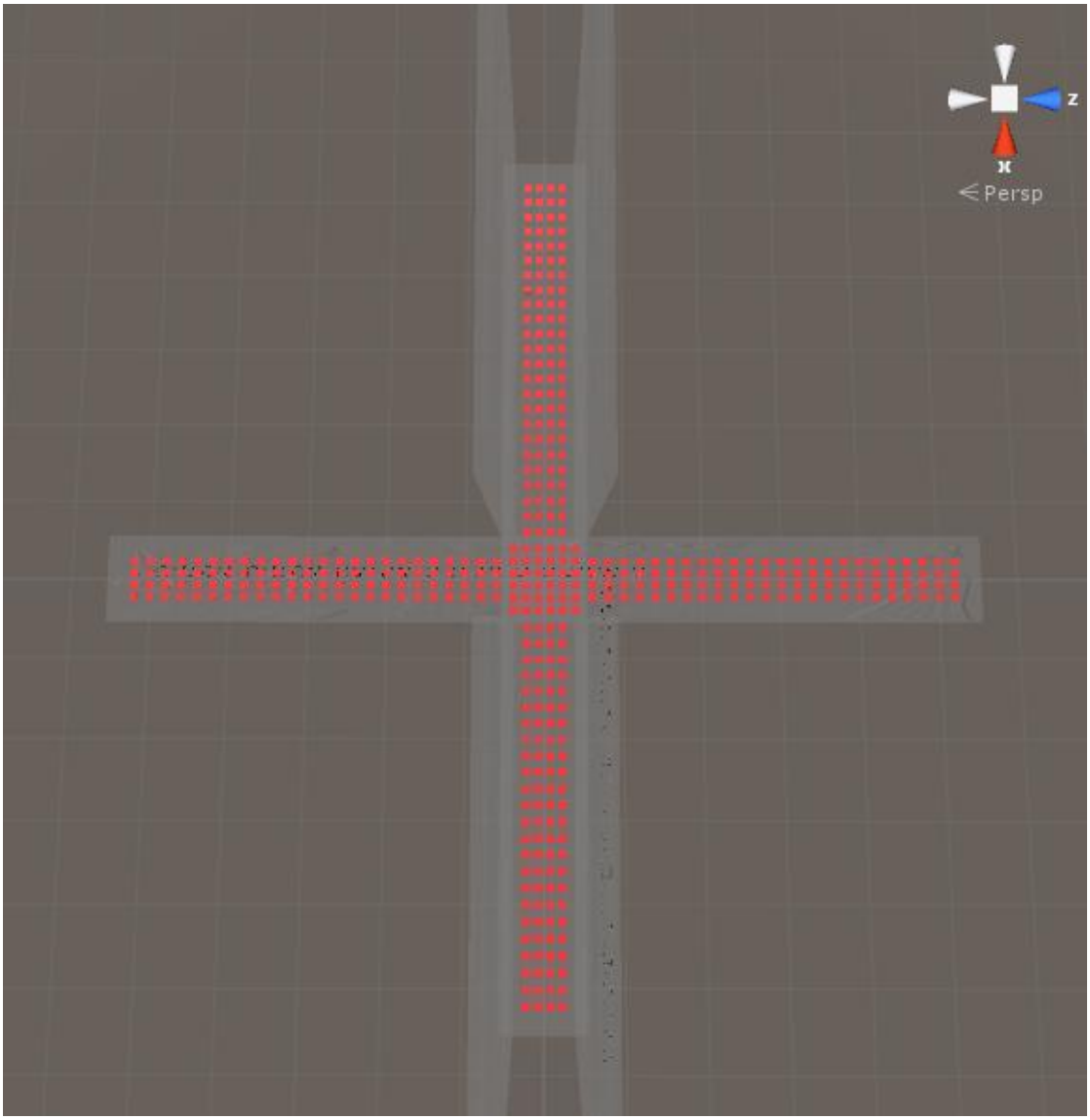


Figura 50: Modelo del laberinto elevado en cruz (color gris) y sus sensores (color rojo) en Unity.

6.1.3. Integración de los modelos 3D

Para lograr desplegar de manera visual que los sensores están siendo tocados por la rata, se agregaron *colisionadores* a los sensores y a algunas partes de la rata (cola, torso, patas y cabeza). Un *colisionador* permite detectar cuando entró en contacto físico con otro. Por lo que la implementación de los *colisionadores* permitió facilitar la programación en el ambiente de Unity mediante el lenguaje de C#. En la Figura 51 podemos observar los *colisionadores* implementados en el modelo del roedor.

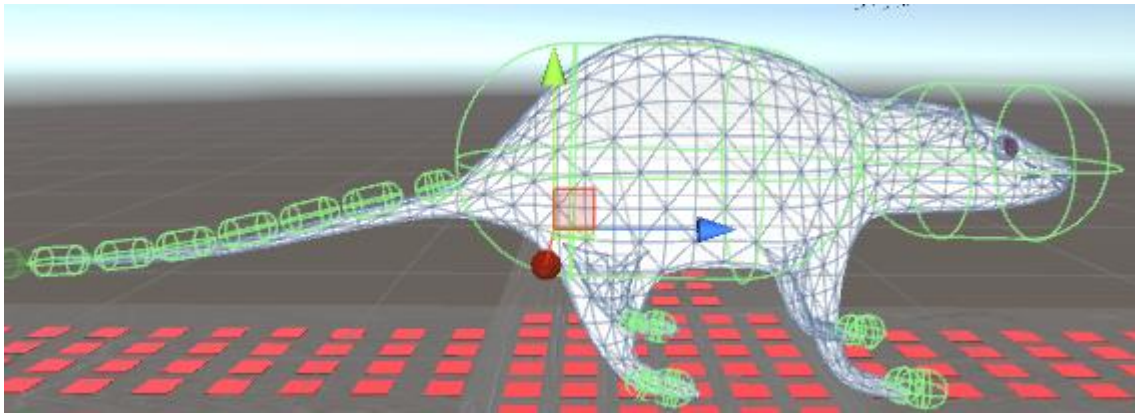


Figura 51: Colisionadores (color verde) de la rata.

Una vez que se realizó la integración de la rata, el laberinto y los sensores con sus respectivos *colisionadores*, se implementó en el lenguaje C# el cambio de animación del roedor mediante el manejo del teclado la computadora y también se implementó el cambio de color de los sensores cuando son tocados por el roedor (a partir de sus *colisionadores*). El resultado de la integración se puede observar en la Figura 52.

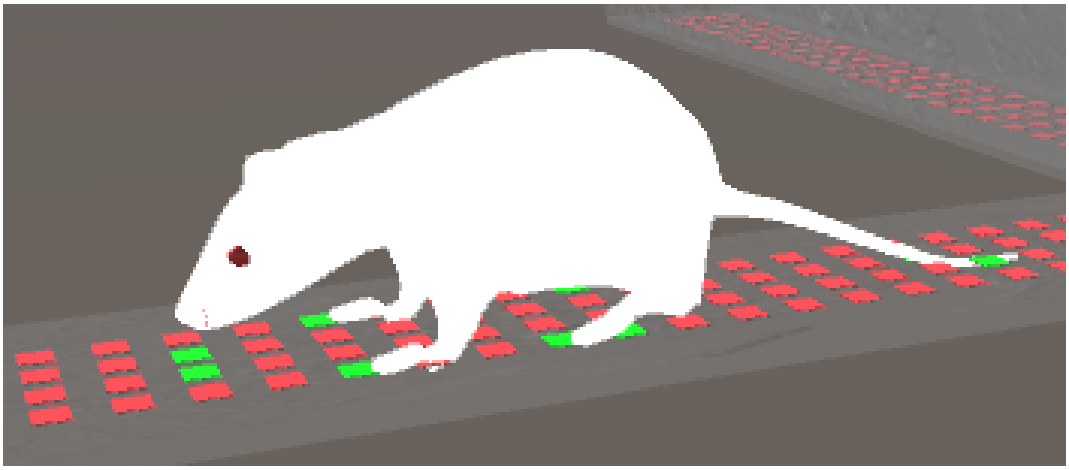


Figura 52: Activación (en verde) de los sensores mediante los colisionadores del roedor.

La simulación permitió generar una herramienta para crear diferentes distribuciones, formas y tamaños de sensores sobre el laberinto y tener una propuesta de diseño de sensores.

6.1.4. Resultado de la simulación

Como resultado de distintas propuestas de sensores se obtuvo la distribución óptima desplegada en la Figura 50. Para el centro del laberinto se diseñaron sensores de 1 x 1 cm, dispuestos en una matriz de 6 x 6, separados por 0.5cm, como se muestra en la Figura 53. Para los brazos (Figura 54) los sensores de 1 x 1 cm se distribuyeron en una matriz de 4 x 24 separados por 1 cm sobre la dirección de los 24 sensores y 0.33 cm sobre la dirección de los 4 sensores. El espacio restante se dejó sin sensores para colocar los soportes para sujetar a la base de acrílico las placas de circuitos conteniendo los sensores y para pasar los cables de alimentación y comunicación. Sin embargo, esta disposición solo es una aproximación de cómo pueden ser distribuidos los sensores.

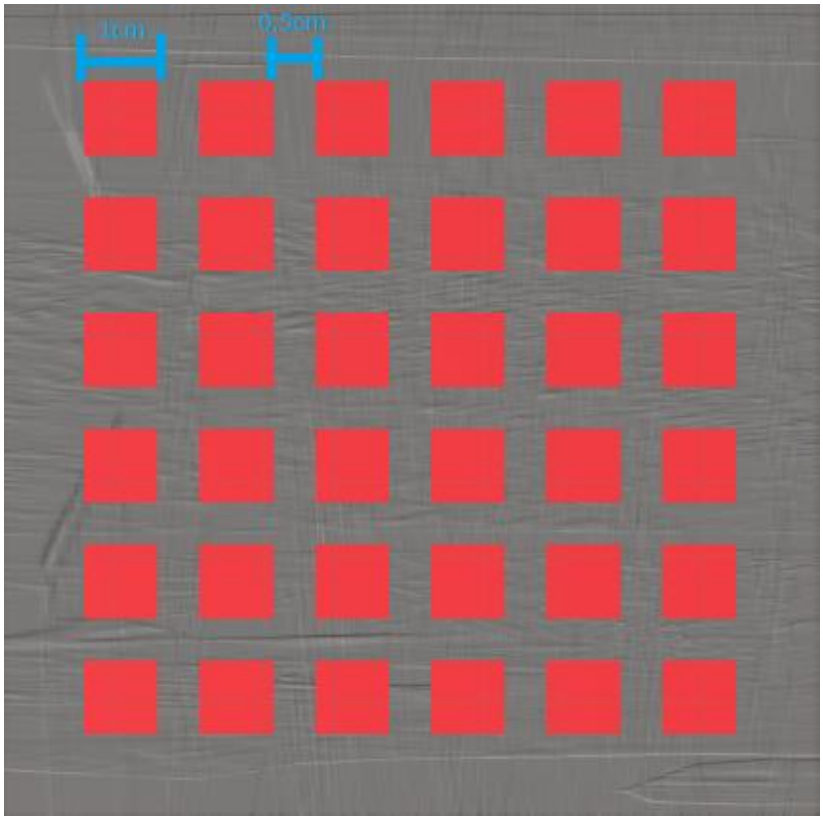


Figura 53: Distribución de sensores en el centro de acuerdo a la simulación.

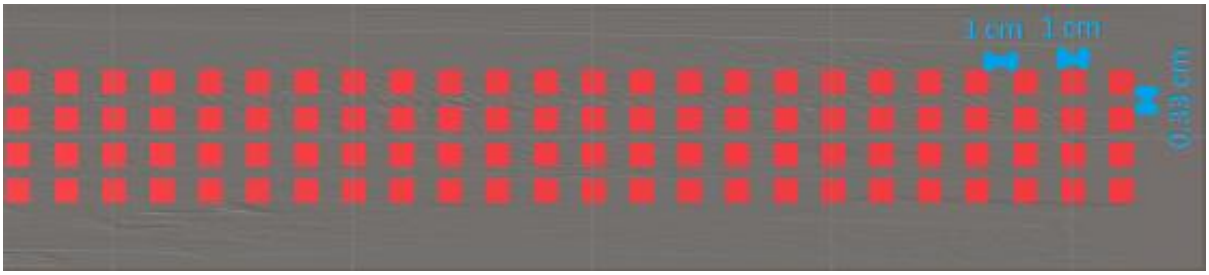


Figura 54: Distribución de los sensores en un brazo de acuerdo a la simulación.

6.2. Selección de controlador de sensores capacitivos

Una vez diseñada la distribución de los sensores, se hizo una búsqueda de controladores de sensores capacitivos de varios fabricantes y se evaluaron las características de cada uno de ellos.

Debido a que la matriz de sensores requiere detectar múltiples puntos de contacto sobre la superficie del laberinto, el controlador debe ser capaz de manejar por separado la información de todos los sensores, por lo que necesariamente debe usar la tecnología *mutual-capacitance*. Puesto que es muy complicado fabricar una sola placa en forma de cruz del tamaño del laberinto que contenga todos los sensores, cada brazo se dividió en dos zonas y el centro en una zona, para un total de 9 zonas, como se muestra en la Figura 55. Lo anterior obligó a buscar un controlador cuyo protocolo de comunicación tuviese la capacidad de manejar un identificador único por cada placa o zona, ya sea un número (I^2C) o una señal digital (SPI) que seleccione al controlador para establecer la comunicación. Con estas restricciones, la búsqueda se redujo a 2 fabricantes: Atmel® y Azoteq®. En la Tabla 5 se muestra una comparación de las características de los controladores de sensores capacitivos de los fabricantes mencionados.

Tabla 5: Comparación de características de tecnología capacitiva de Atmel y Azoteq

Fabricante	**I^2C**	**SPI**	**Mutual-capacitance**
Atmel	Si	Si	Si
Azoteq	Si	No	Si

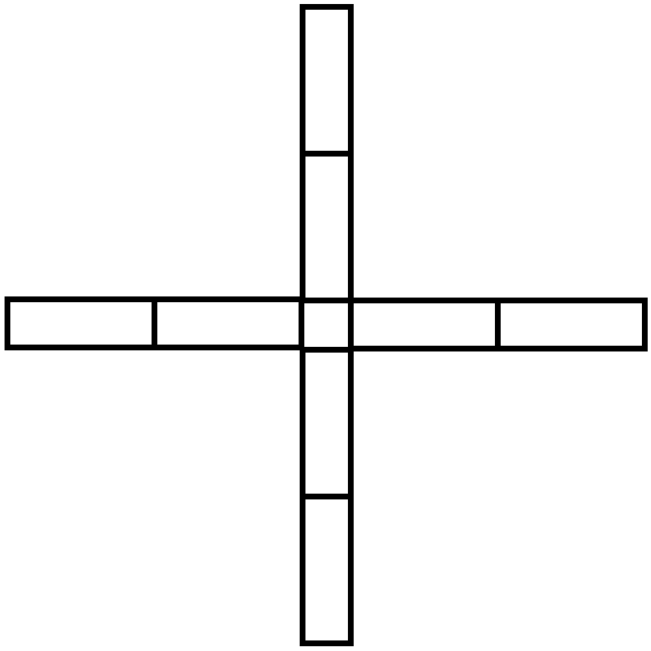


Figura 55: Esquema de la distribución de las placas con las matrices de sensores capacitivos en el laberinto

El resultado de la comparación de las tecnologías, nos permitió seleccionar al fabricante Azoteq®, ya que se encontró la ventaja del protocolo I^2C, ya que permite reducir la cantidad de cables conectores que cruzarán a través de todo el laberinto. Aunque la tecnología de Atmel® también soporta el protocolo I^2C, la mayoría de sus controladores están limitados a un máximo de 4 direcciones esclavo (identificadores) diferentes, por lo que en las mismas señales de comunicación solo pueden comunicarse como máximo este número de dispositivos. Por el contrario, los controladores de Azoteq® cuentan con una herramienta que permite cambiar los 7 bits dirección de esclavo de cada controlador, haciendo posible generar hasta 128 direcciones esclavo diferentes.

Los controladores de Azoteq® que cuentan con las características mencionadas anteriormente son de la serie IQS5xx-B000, de los que existen tres versiones: IQS525-B000, IQS572-B000 e IQS550-B000, capaces de controlar 25, 72 y 150 canales, respectivamente.

6.3. Descripción del controlador IQS5xx-B000

El dispositivo IQS5xx-B000 es un controlador de sensado capacitivo *mutual* con salidas de proximidad, toques, *snap* (tecnología patentada de Azoteq®), trackpads y gestos. Las principales características que podemos encontrar de este controlador son:

- Proximidad, toque y *snap* en cada canal.
- Soporta hasta 5 multi-toques.
- Gestos de uno y muchos dedos.
- Máxima resolución de 3584x2304 (IQS550).
- Interfaz de comunicación I^2C.
- ATI: Ajuste automático para una sensibilidad óptima.
- Voltaje de alimentación: 1.65 a 3.6V.
- Detección y eliminación de ruido.

En la Tabla 6 se enlistan las características de las tres versiones del controlador IQS5xx-B000.

Tabla 6: Características de las versiones del controlador IQS5xx-B000.

	IQS550	IQS572	IQS525
Máximo número de canales	150	72	25
Frecuencia de reporte	100 Hz	135 Hz	190 Hz

Máxima resolución (Tx Rx)	3584 x 2304 (15 x 10)	2048 x 1792 (9 x 8)	1280 x 768 (6 x 4)

Para poder realizar la configuración o lectura de las salidas de los canales es importante comprender la interfaz de comunicación del controlador. Como se enlista en las características, el dispositivo se comunica mediante el protocolo I^2C; sin embargo, cuenta con una línea adicional de comunicación denominada RDY (*ready* en inglés, o listo traducido al castellano). Esta línea adicional es una señal digital de salida que, cuando su estado lógico se encuentra en “1”, indica que hay datos con información nueva disponible acerca de los canales de los sensores. Esta característica permite que un dispositivo maestro no realice lecturas innecesarias o repetidas al controlador, óptimizando la comunicación. Esta señal digital es independiente para cada dispositivo, es decir, que no se puede compartir de la misma manera que se comparten el SDA y SCL del protocolo I^2C. Sin embargo, se puede establecer la comunicación de manera forzada sin el uso de esta señal siguiendo la lógica que se muestra en el diagrama de la Figura 56 y activando el modo evento del controlador. Una vez establecida la comunicación el microcontrolador maestro puede realizar las operaciones de lectura o escritura siguiendo el protocolo estándar I^2C.

Para poder configurar o leer información del controlador, se manejan registros internos de dicho dispositivo. Cada registro tiene una dirección de 16 bits en el mapa de memoria del controlador, por lo que se debe indicar la dirección para poder leer o escribir en los registros. En caso de que la dirección no sea establecida, el controlador utiliza una dirección por defecto para leer o escribir, la cual también es configurable. Cuando se inicia una lectura o escritura, por cada byte que sea leído o escrito, el controlador incrementa internamente la dirección en uno para que la operación próxima se realice a la siguiente dirección de memoria. Para comprender mejor estas operaciones, en la Figura 57 se ilustra el proceso de escritura y en la Figura 58 el de lectura.

Cuando se ha terminado de leer o escribir información, la comunicación no se termina con un simple bit de Stop, para ello se requiere que el dispositivo maestro escriba cualquier byte en la dirección de memoria 0xEEEE para cerrar la comunicación adecuadamente. Finalizando la comunicación, el controlador continúa con el procesamiento de sensado.

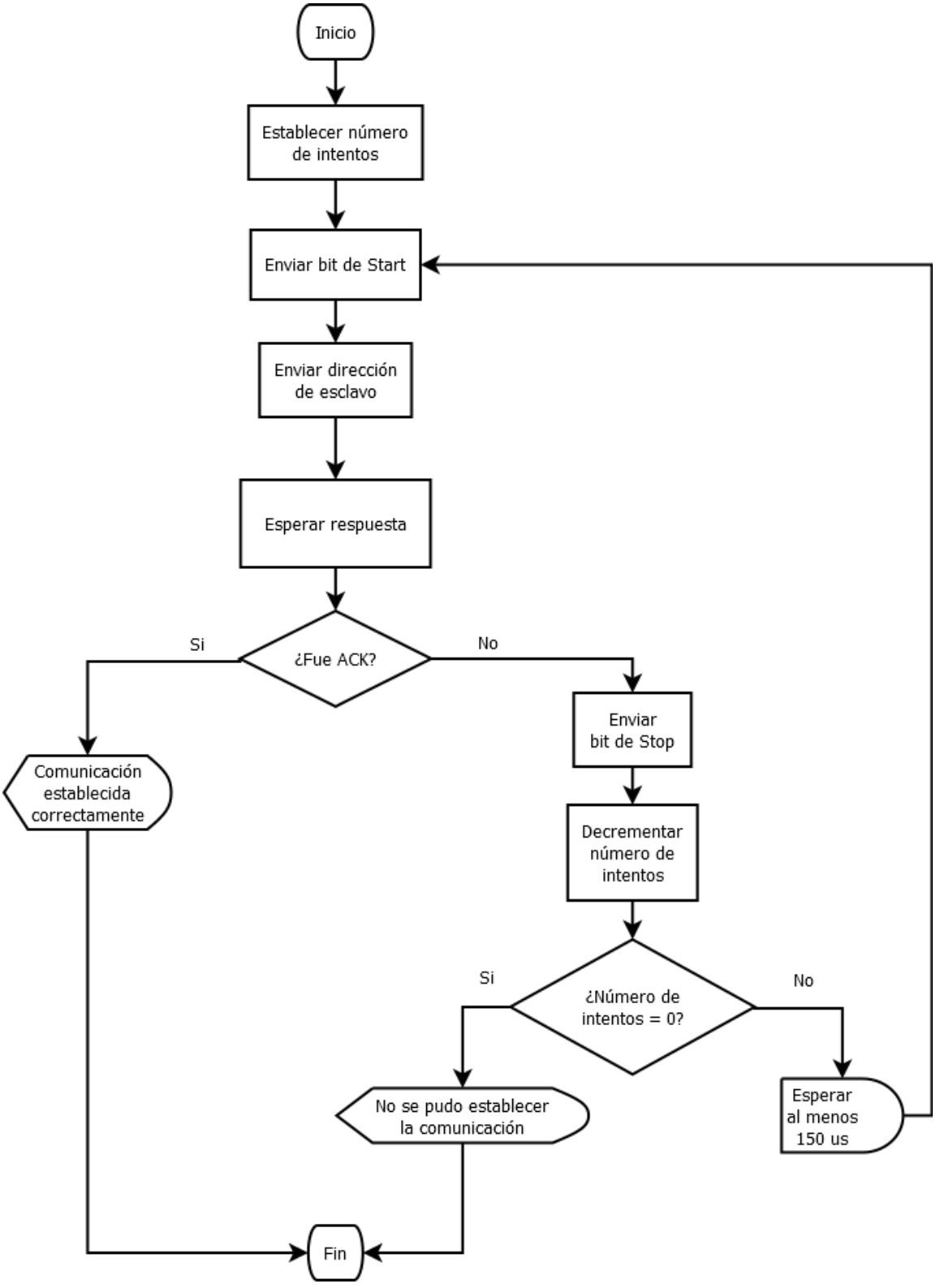


Figura 56: Inicio de comunicación forzada sin el uso de RDY

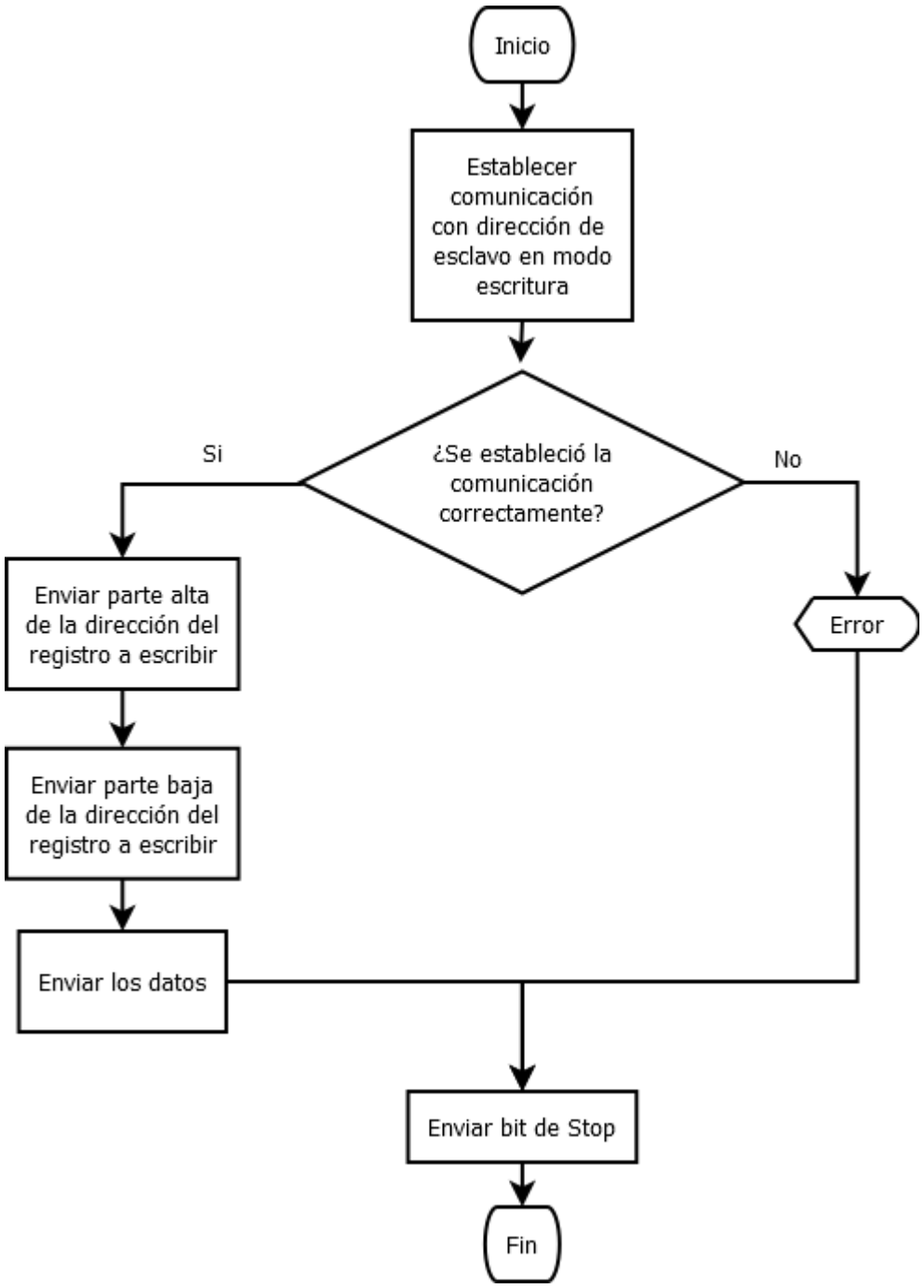


Figura 57: Proceso de escritura para el IQS

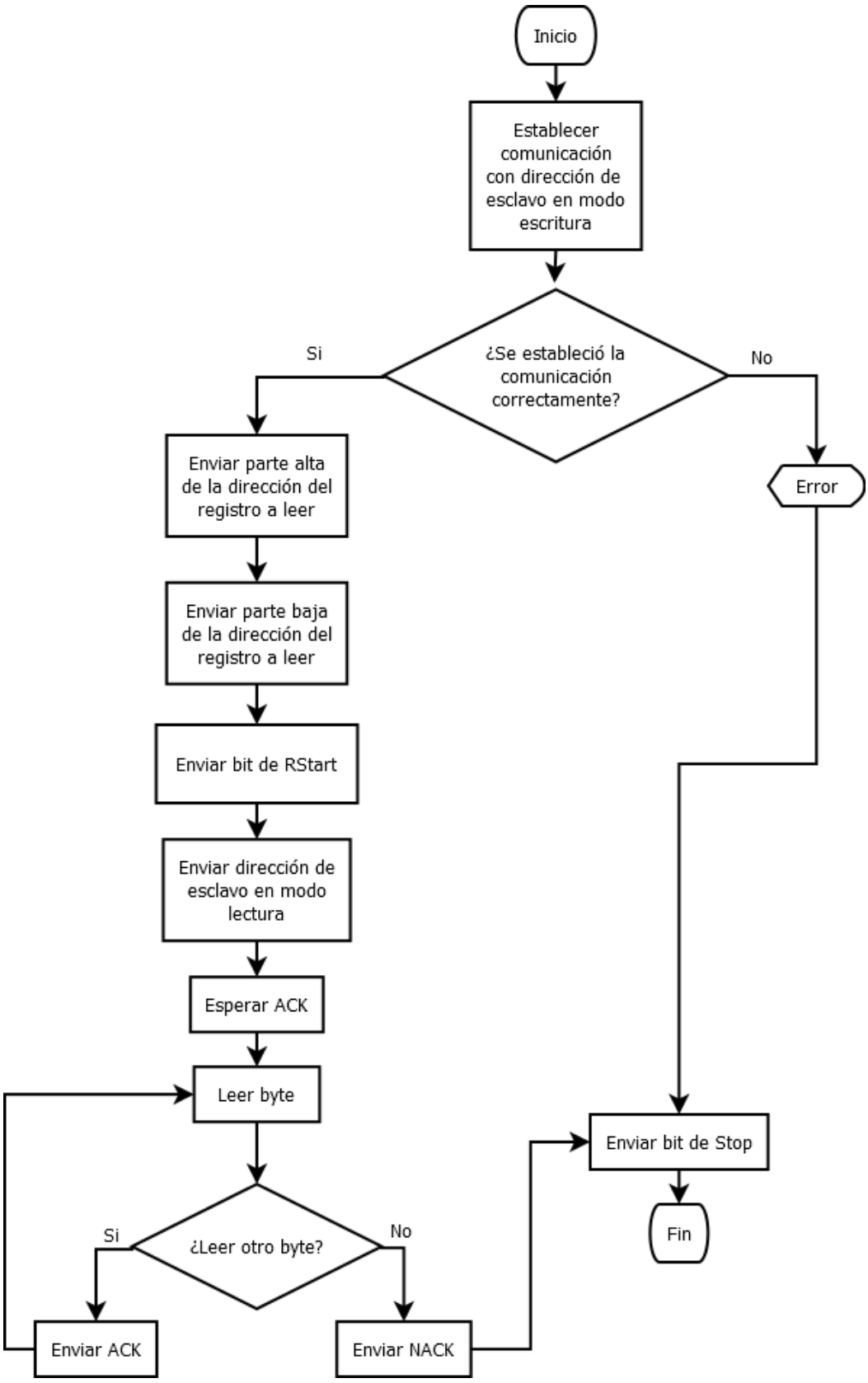


Figura 58: Proceso de lectura para el IQS

Los registros del controlador que son de interés se describen a continuación. Por convención los nombres de los registros se escribirán en MAYÚSCULAS y los nombres de los bits en minúscula y *cursiva*.

El registro SYSTEM_INFO_0 almacena información general del controlador. Este registro cuenta con una bandera denominada *show_reset*. Cuando la bandera está en "1" indica que el dispositivo fue reiniciado. Este registro es solo de lectura, por lo que, para poder bajar o desactivar esta bandera, es necesario escribir el valor de "1" en la bandera denominada *ack_reset* del registro SYSTEM_CONTROL_0.

Otra bandera que es importante considerar, sobre todo cuando el dispositivo es reiniciado o es su primer inicio (o encendido), es la bandera *setup_complete*, que se encuentra en el registro SYSTEM_CONFIG_0. Esta bandera sirve para indicar que el dispositivo se ha terminado de configurar y el controlador comience con su labor de sensado. Para ello es necesario escribir "1" en esta bandera.

En el registro SYSTEM_CONFIG_1 se encuentra la bandera event_mode, que activa o desactiva el modo evento del controlador, esta bandera es importante para poder iniciar el modo forzado de la comunicación I^2C. Generalmente la bandera es configurada usando la interfaz gráfica que provee Azoteq®, ya que se utiliza la comunicación mediante la señal RDY.

Las configuraciones del controlador permiten definir (generalmente mediante la interfaz gráfica) el orden de las señales Tx y Rx de los canales. Es decir, se pueden intercambiar el orden de las filas (Tx) y también el orden de las columnas (Rx) de acuerdo a como se encuentren físicamente distribuidas en la matriz de sensores. Al igual, permite desactivar canales, es decir, intersecciones entre Rx y Tx. Esta configuración es guardada en registros del controlador.

La información de la cantidad de transmisores Tx que están siendo utilizadas es almacenada en un registro denominado TOTAL_TX, de manera análoga se almacena la información de los receptores Rx en el registro TOTAL_RX. Con esta información se puede conocer la dimensión de la matriz de sensores.

Para saber si un canal está habilitado, de acuerdo a la matriz formada por los sensores, el controlador guarda dicha información en el conjunto de registros ACTIVE_CHANNELS que están conformados de 30 bytes consecutivos en la memoria. Dado que la máxima dimensión de la matriz que se puede implementar con los dispositivos IQS es 15x10, cada par de bytes contienen la información de estados de cada fila, de un total de 15 filas, mientras que cada bit de cada par de bytes indica el estado de cada columna, de un total de 10 columnas, por lo que solo 10 bits almacenan información útil de cada par mientras que los 6 bits restantes no tienen algún significado. La Tabla 7 indica la organización de los 30 bytes en memoria, donde "X" representa la dirección de cada byte y en cada par de bytes consecutivos se almacena la información del estado de una fila (parte alta y baja). La Tabla 8 indica la información de las columnas almacenada en cada par de bytes o fila. De la misma manera se guarda la información de proximidad y toque, PROX_STATUS y TOUCH_STATUS, respectivamente. Donde la proximidad es indicativa de que un objeto esta próximo al canal y el toque indica que un objeto tocó el canal. Por lo que también puede ser representadas con la Tabla 7 y la Tabla 8.

Con la información del conjunto de registros descritos con anterioridad es posible representar el estado de la matriz de sensores.

Otro registro útil es el denominado NUMBER_OF_FINGERS que indica el número de objetos presentes sobre la matriz de sensores. Con este registro es posible detectar si hay algún objeto o no sobre la superficie.

Tabla 7: Representación del conjunto de bytes que indican los estados (activos, proximidad o toque) por filas, donde la "X" representa la dirección en memoria de cada byte.

Dirección	Byte
X	Byte alto de la Fila0
X+1	Byte bajo de la Fila0
X+2	Byte alto de la Fila1

X+3	Byte bajo de la Fila1
…	…
X+28	Byte alto de la Fila14
X+29	Byte alto de la Fila14

Tabla 8: Representación de las columnas por cada bit en cada par de bytes que indican los estados de cada canal (activos, proximidad o toque) en una fila.

	Byte Alto						**Byte Bajo**									
Colum-nas	-	-	-	-	-	-	Col 9	Col 8	Col 7	Col 6	Col 5	Col 4	Col 3	Col 2	Col 1	Col 0
Fila	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

6.4. Diseño de los sensores capacitivos

Una vez seleccionado el controlador, se definió el diseño de los sensores capacitivos a partir de éste. Para ello se siguieron los criterios y recomendaciones que el proveedor propone en su documentación.

Para diseñar los sensores se consideraron las siguientes restricciones:

- En la zona central existe un espacio disponible de 9.06 x 9.06 cm para distribuir sensores.
- En los brazos se tiene un espacio disponible de 6 x 43 cm para distribuir sensores.
- En la simulación 3D se obtuvo que pueden colocarse hasta 6 x 6 sensores.
- El grosor del acrílico con el que está fabricado el laberinto es de 5 mm.

- La colocación de los sensores capacitivos por debajo del acrílico los protege de los líquidos, como la orina que las ratas suelen dejar sobre la superficie del laberinto y la solución de etanol usada para limpiar el laberinto entre cada prueba.

La serie de controladores IQS5xx-B000 para sensado capacitivo se especializa en el control de pantallas táctiles y también para el control de trackpads. De acuerdo al fabricante, el diseño de trackpads utiliza un patrón de diamantes. Sin embargo, debido a que el grosor del acrílico que actúa como panel frontal es de 5 mm, el diseño basado en los criterios de trackpad no es apropiado, ya que la documentación recomienda que el panel frontal tenga un grosor máximo de 2 mm. Por lo anterior, se optó por un diseño de botones siguiendo los ejemplos validados para paneles de 5 mm de grosor que provee Azoteq® en el apartado 5.2.1.

En la Tabla 3 se puede ver que los diseños 5 y 6 son los que tienen una mejor confiabilidad para superficies de 5 mm, además de que ambos tienen un buen nivel de inmunidad a los líquidos. Sin embargo, debido a las restricciones del área disponible para colocar sensores, el diseño 5 es el que mejor se adapta ya que es el de menores dimensiones (10.4 x 8.4 mm).

Como se observa en la Figura 59, utilizando el diseño 5, se lograron colocar (6 x 7) – (4) sensores. Se removieron los sensores de las esquinas debido a que en una de ellas se colocaron los componentes necesarios para su correcto funcionamiento. Por el número de sensores necesarios en la zona central (42) el controlador que se utilizó fue el IQS572-B000.

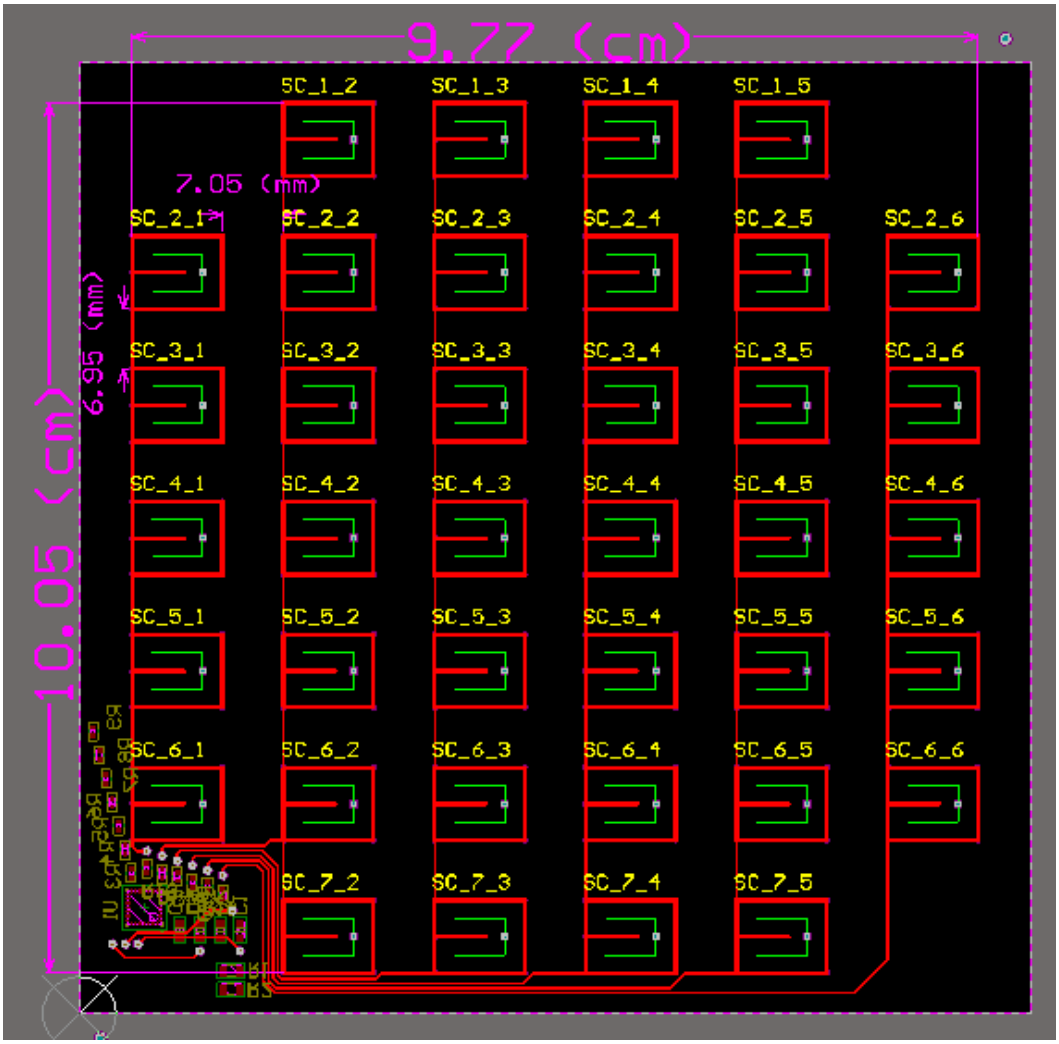


Figura 59: Diseño de sensores de la zona central

El resultado del diseño electrónico de la placa de la zona central se observa en la Figura 60.

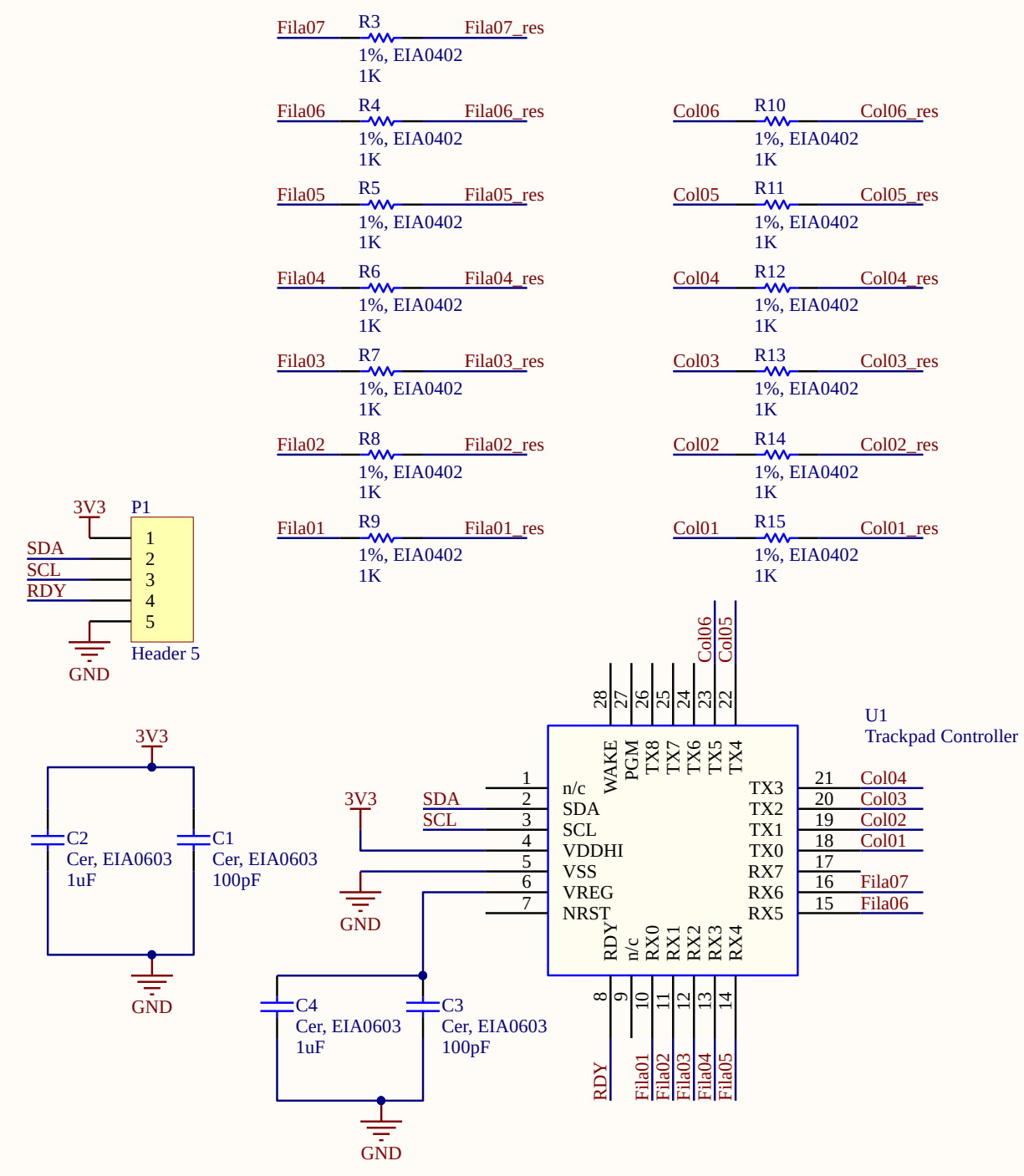


Figura 60: Esquema del circuito de la zona central

En la Figura 61 podemos observar el resultado de la manufactura de los sensores de la zona central.

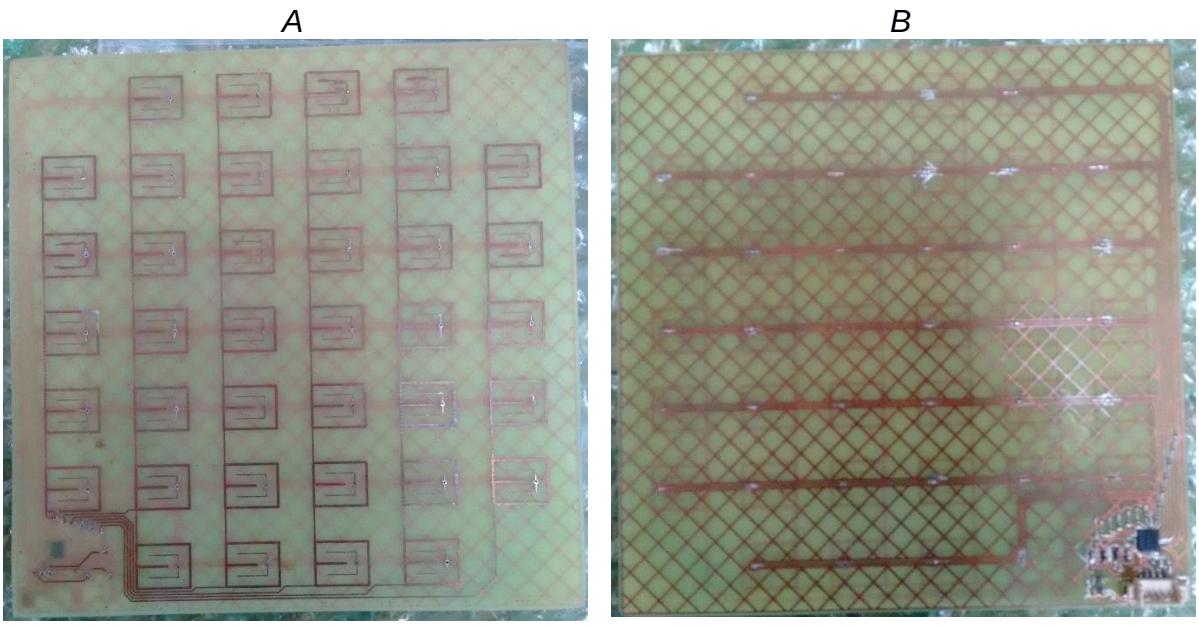


Figura 61: Vista frontal (A) y vista trasera (B) de la zona central manufacturada

Para comprobar el funcionamiento de la placa se utilizó el DS100 de Azoteq®, que es una herramienta que permite configurar y leer la información de los sensores por el protocolo I^2C, mediante una conexión USB en la computadora y un software propietario (Figura 62). Usando esta herramienta se realizó la configuración y calibración de los sensores.

Con el fin de verificar el funcionamiento de los sensores en un escenario real, la placa de sensores se situó por debajo de la zona central del laberinto y se colocó encima una rata. Durante la prueba se notó la activación de los sensores cuando el animal pasaba por el centro del laberinto. Por ejemplo, en la Figura 63 se logra apreciar la activación de los sensores por la cola del roedor.

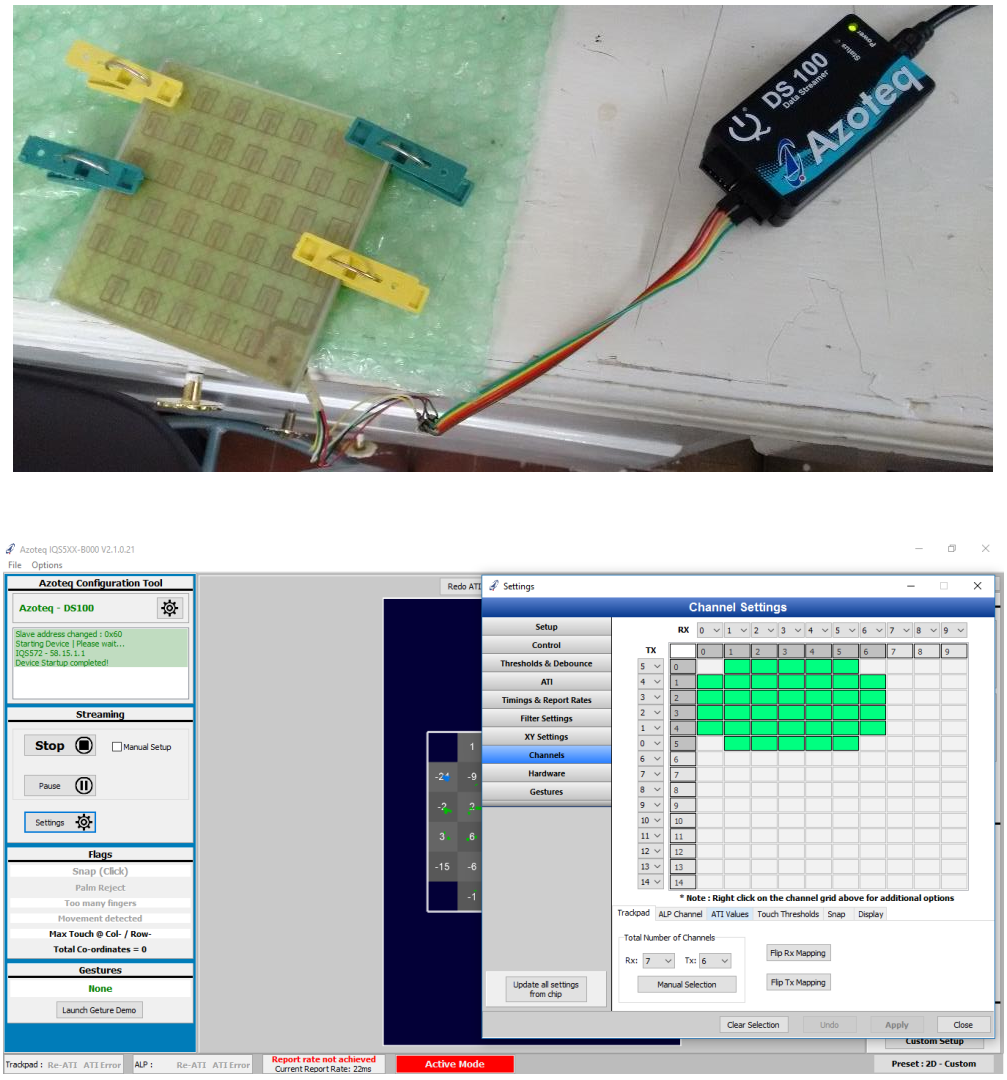


Figura 62: Configuración del controlador de sensores

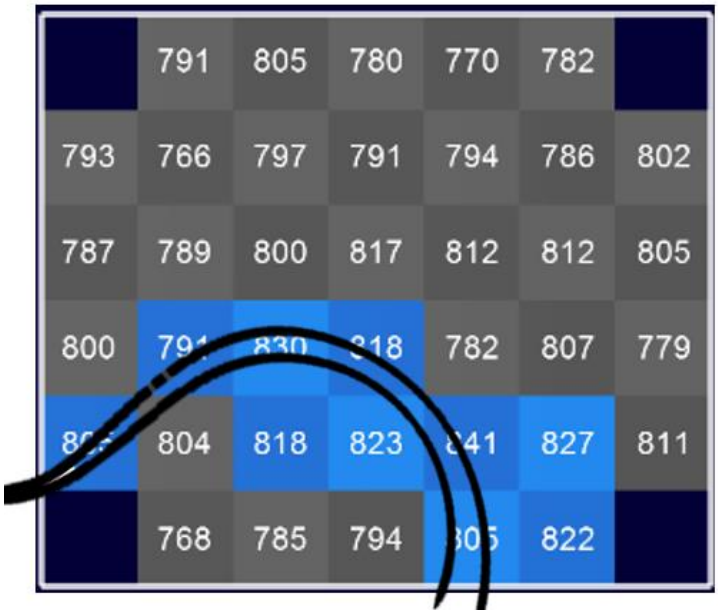


Figura 63: Activación de los sensores de la zona central por la cola de la rata

En la Figura 64 se puede observar la distribución de los sensores para una placa del brazo (la mitad del brazo del laberinto). En cada placa del brazo se colocaron (4 x 14) - (1) sensores. Se eliminó un sensor porque en ese espacio se colocaron todos los componentes necesarios para el funcionamiento de los sensores de la placa.

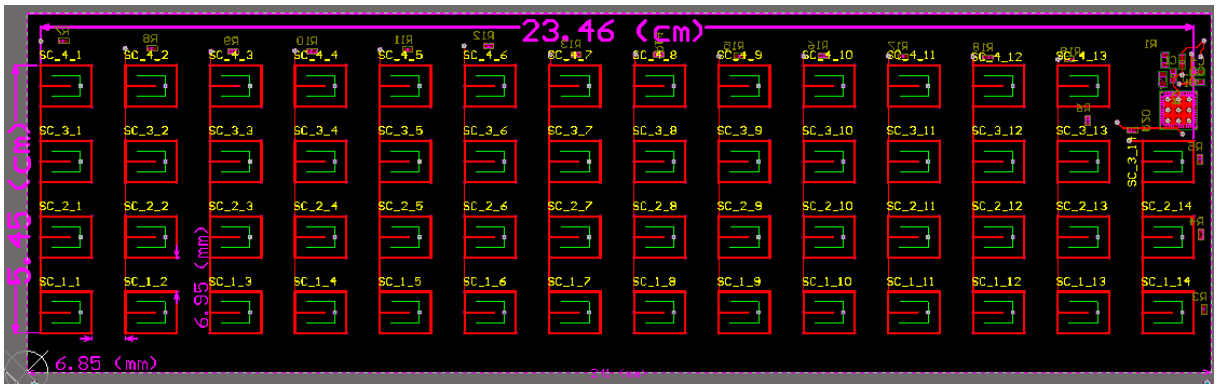


Figura 64: Distribución de los sensores de una placa del brazo del laberinto

Debido a que se requieren monitorear 55 sensores por cada placa, se utilizó el controlador IQS550-B000. En la Figura 65 se ilustra un esquema del diseño electrónico

de esta placa, y en la Figura 66 se observa su funcionamiento, mediante la activación de los sensores por dos dedos de una mano humana.

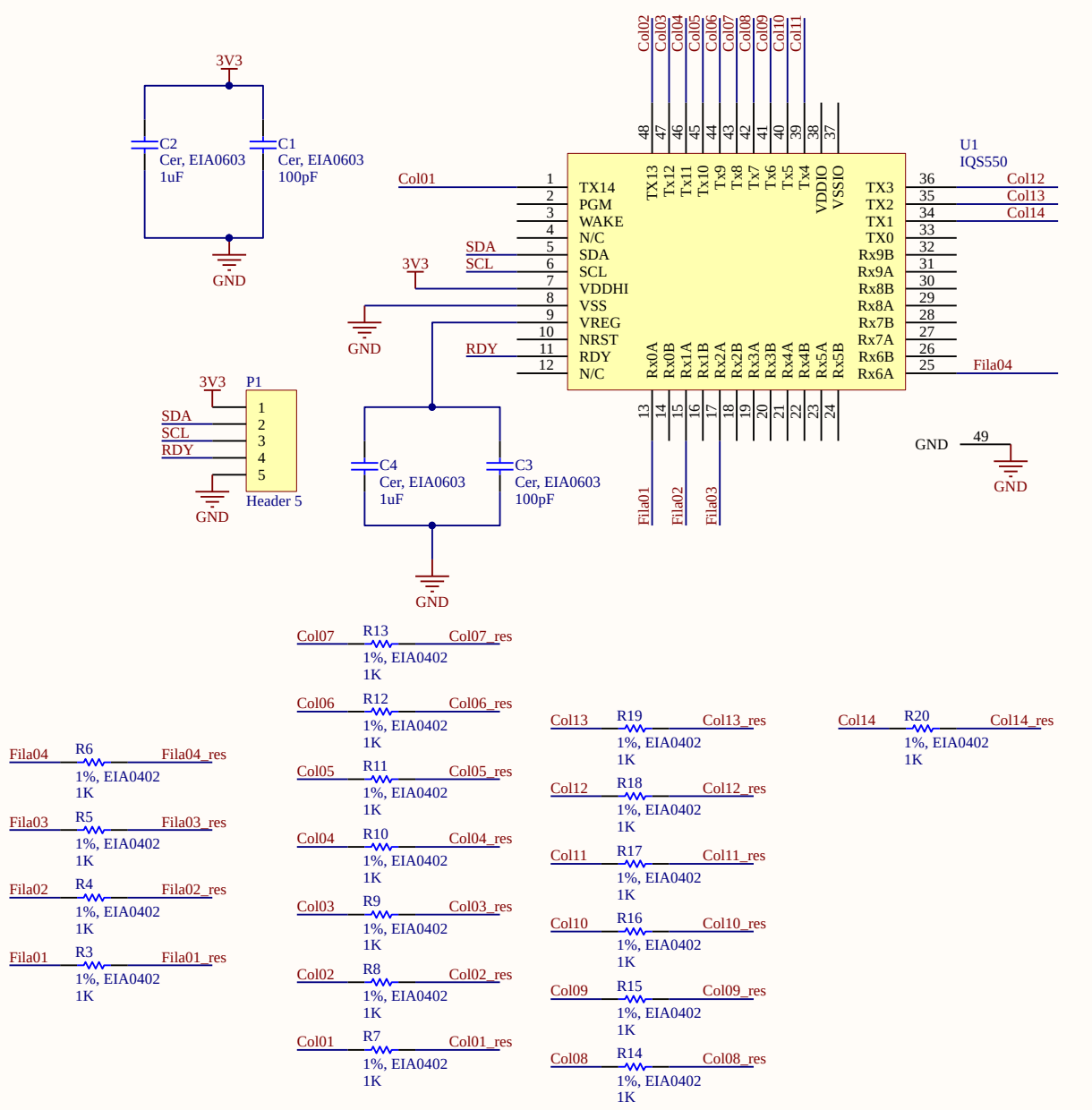


Figura 65: Esquema del circuito de los sensores del brazo

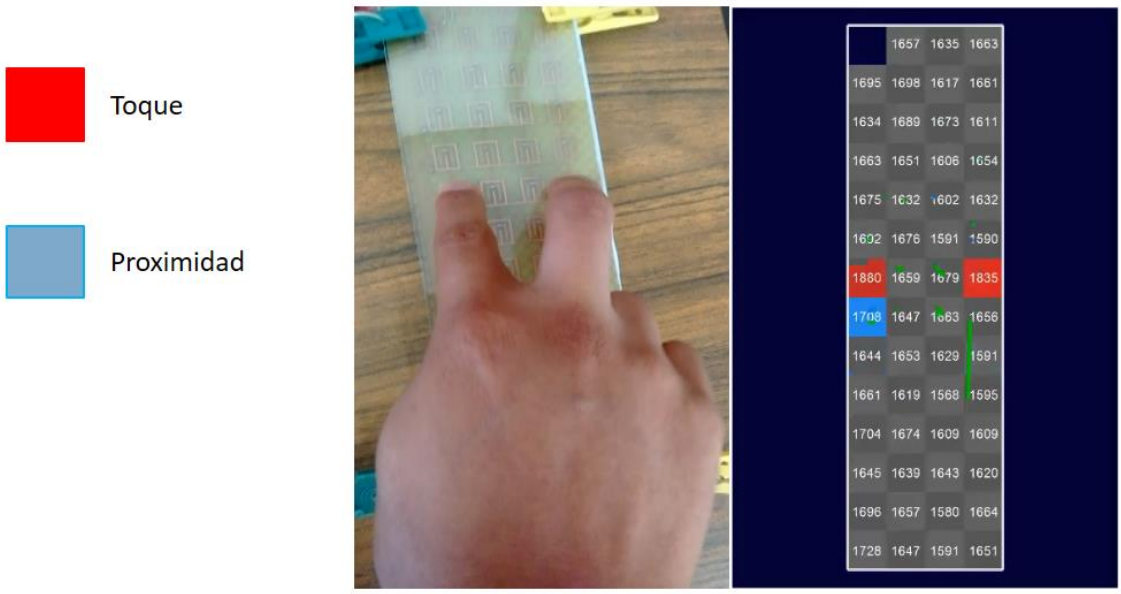


Figura 66: Prueba de funcionamiento de los sensores del brazo mediante una mano

6.5. Integración de los sensores al laberinto

Una vez validado el funcionamiento del diseño de la matriz de sensores del laberinto, se procedió a realizar la manufactura de todas las placas de sensores necesarias. Pero para ello fue necesario realizar algunas modificaciones al laberinto para poder acomodar las placas, sujetarlas y mantenerlas fijas.

Se le agregó un contorno de acrílico de 1 cm de altura a todo el laberinto para sujetar una tapa inferior (copia de la cruz de acrílico) y unas pestañas laterales para sujetar las paredes desmontables (Figura 67).

Figura 67: Vista lateral del laberinto automatizado

Para mantener en posición la placa de sensores y la cablería, se elaboró una tapa inferior de acrílico de las mismas dimensiones que la cruz de acrílico, añadiéndoles unas varillas de acrílico de 1 x 1 cm de corte transversal, con el propósito de hacer presión a la esponja (foam) que mantiene fijas a las placas (Figura 68). En la Figura 68 también se ilustran las varillas colocadas en todas las esquinas de la cruz, en cuyo centro se colocaron tuercas para poder atornillar la tapa y mantenerla en posición.

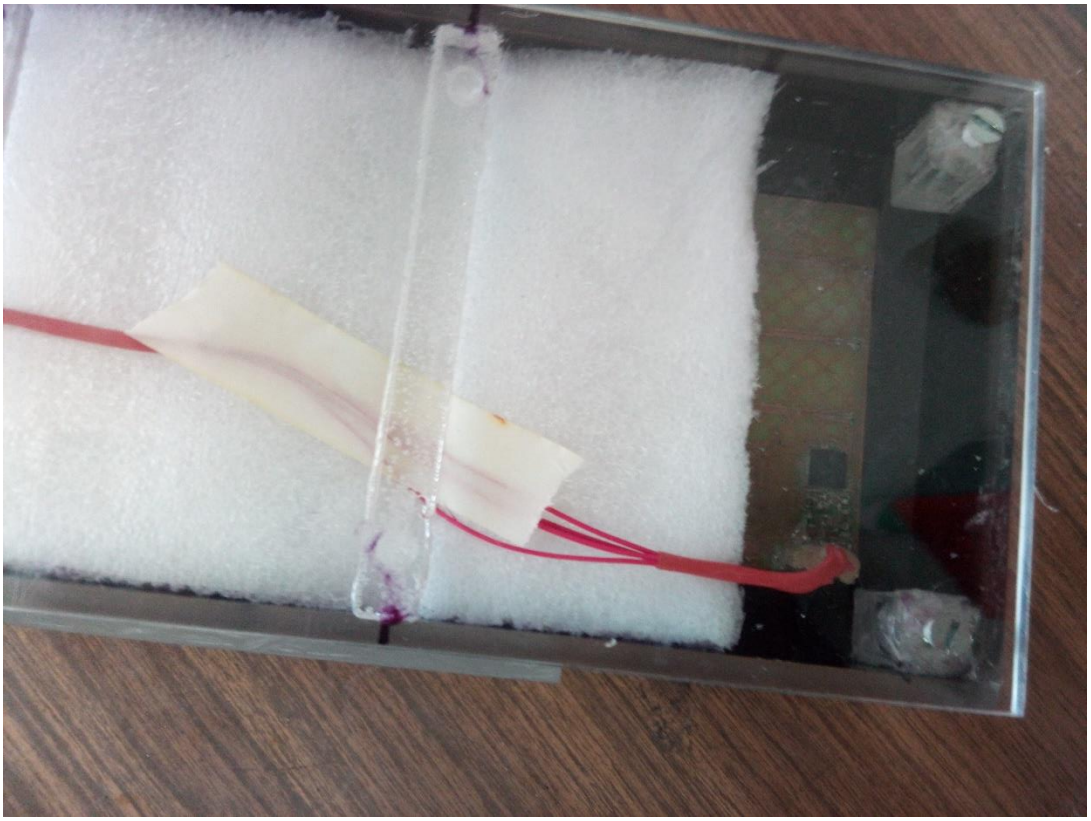


Figura 68: Vista inferior del laberinto ilustrando el foam, la varilla transversal y las varillas para sujetar la tapa.

Para lograr la comunicación con todas las placas se requiere que todas estén interconectadas en el mismo bus de comunicación y que compartan la misma fuente de energía (3.3V y GND). Para ello se diseñaron placas puente que interconectan las líneas que provienen de las placas adyacentes (Figura 69). Finalmente se colocó una salida del cable que transmite las señales de todas las placas, lo cual se puede observar en el montaje final de los sensores ilustrado en la Figura 70.

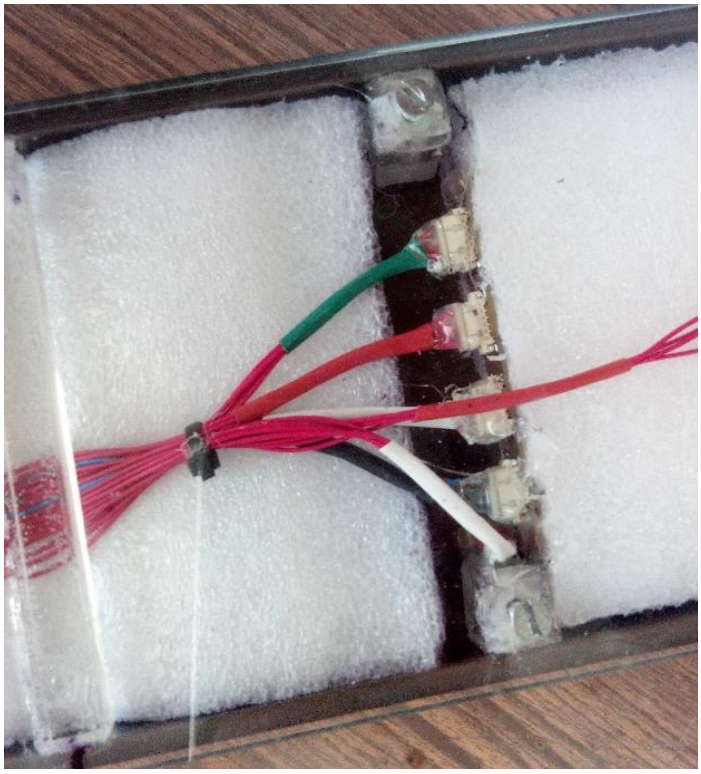


Figura 69: Interconexión de los sensores mediante placas puente

Figura 70: Montaje final de las placas en el laberinto

6.6. Verificación del funcionamiento de todos los sensores

Para validar el funcionamiento de todas las placas se implementó un código en lenguaje ensamblador para leer y/o escribir información por I^2C y transmitirla a una PC por medio del protocolo USB. El firmware se desarrolló para una tarjeta de adquisición de propósito general denominada “Hideyo Balam” (Figura 71) desarrollada en laboratorio de ingeniería biomédica del CIR “Dr. Hideyo Noguchi”, que utiliza el microcontrolador PIC18LF2550 de Microchip®. El firmware es una modificación de un software previamente desarrollado en este laboratorio que permite al microcontrolador

mandar información a la PC a través del protocolo USB o viceversa (Salazar Loría, 2012). Al código original se le agregó la configuración del módulo IIC.

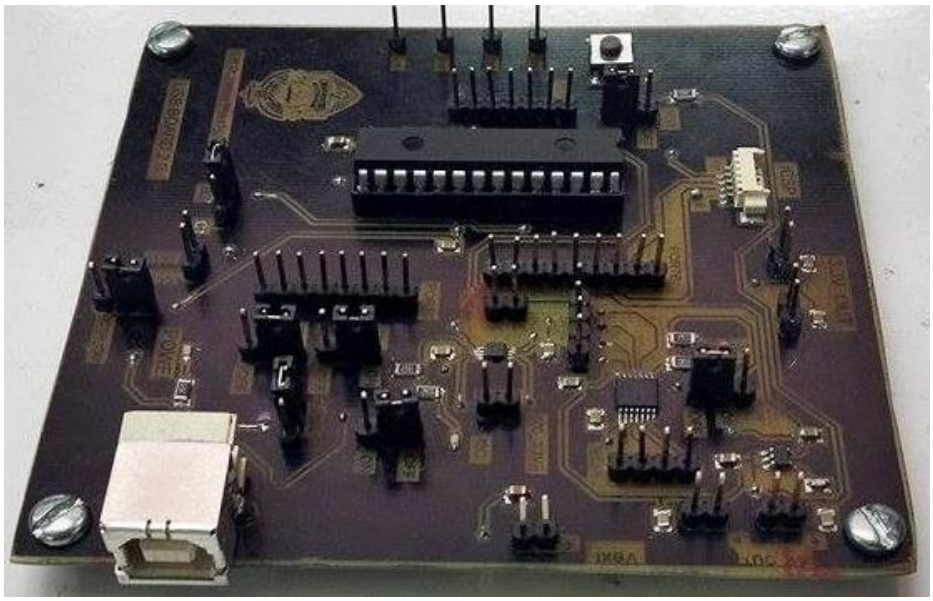


Figura 71: Tarjeta de adquisición de datos "Hideyo Balam"

La lógica principal del firmware desarrollado se muestra en el diagrama de la Figura 72. La comunicación que se realiza entre la PC y la tarjeta “Hideyo Balam” está conformado de un búfer de 64 bytes.

Cuando la PC envía datos, solo los primeros 8 bytes tienen un significado (Tabla 9). El primer byte indica si se requiere inicializar el dispositivo (Modo = 0) para que el dispositivo IQS inicie el proceso de sensado, o si se requiere realizar una lectura de bytes a una dirección específica del dispositivo IQS (Modo = 1). El segundo byte indica la dirección de esclavo del dispositivo IQS al que se le realizará la operación. Cuando Modo = 0, los siguientes tres bytes no importan. En cambio, cuando Modo = 1 el tercer y cuarto byte indican la posición de la memoria interna del IQS se desea leer, y el quinto byte la cantidad de bytes a leer (máximo 63). El sexto byte indica el número de intentos para iniciar la comunicación con el IQS, y el séptimo y octavo byte indican el tiempo de espera entre tiempos expresado en milisegundos.

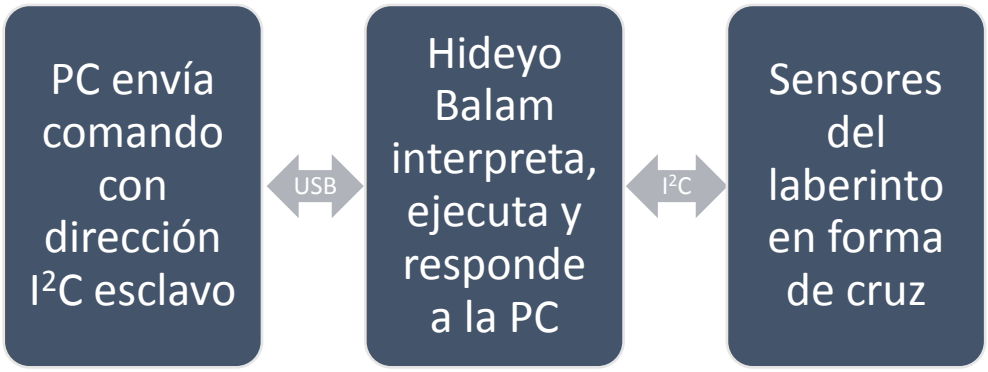


Figura 72: Diagrama de funcionamiento del firmware para leer la información de los sensores

Tabla 9: Significado de los bytes enviados por la PC a la Tarjeta Hideyo

Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
Byte alto del retardo en ms entre intentos	Byte alto del retardo en ms entre intentos	Intentos para iniciar la comuni-cación	Número de bytes a leer	Byte bajo de la dirección del registro del IQS	Byte alto de la dirección del registro del IQS	Dirección I²C	Modo. 0: Inicializa-ción 1: Lectura

La respuesta de la tarjeta “Hideyo Balam” puede estar conformada desde 1 byte hasta un máximo de 64 bytes (el tamaño del búfer). El primer byte indica el resultado de la operación, pudiendo significar lo siguiente:

- 0: Operación exitosa.
- 1: Dispositivo ya inicializado.
- 2: Error de comando.

Cuando la respuesta es 0 y se realizó una lectura, los bytes restantes representan la lectura realizada por el IQS; en caso contrario los demás bytes son ignorados.

También se implementó en la PC una interfaz gráfica para interpretar los datos provenientes de la tarjeta. La interfaz para el diseño y manejo de interfaces gráficas se realizó en el lenguaje C++ y el conjunto de bibliotecas QT usando un sistema operativo GNU/Linux (Figura 73). La interfaz permite guardar imágenes del estado de los sensores en cada segundo transcurrido (Figura 74). También permite programar el tiempo total en el que se realizará la captura de las imágenes de los sensores, simulando de esta manera su activación a lo largo del tiempo de experimentación. Igualmente es posible seleccionar las placas de las que se recolectará la información. La interfaz también es capaz de generar un archivo en formato JSON que contiene la información de los sensores por cada segundo. También permite importar un archivo JSON y generar las imágenes a partir de la información contenida en el archivo.

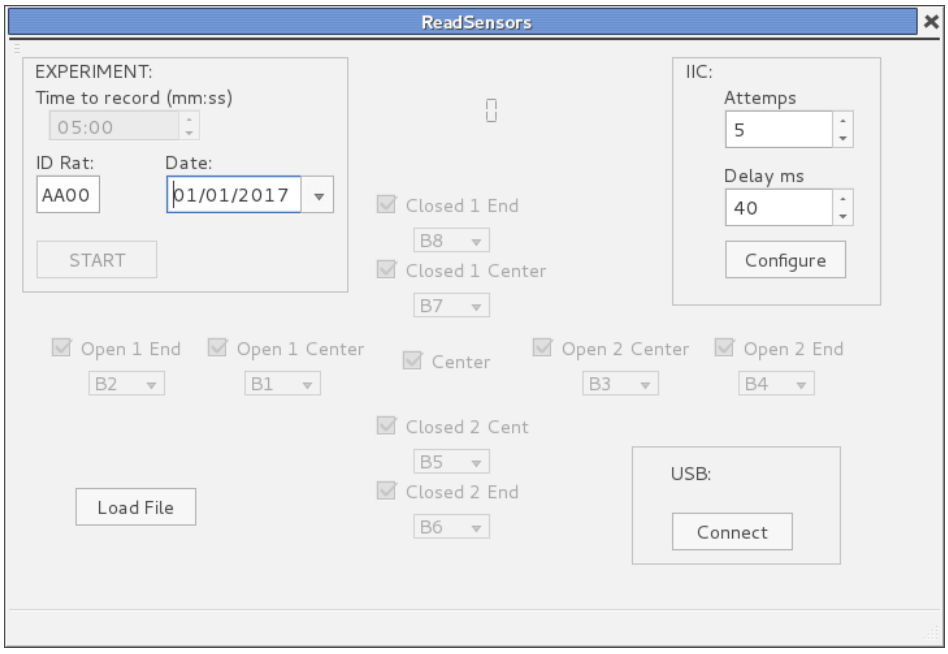


Figura 73: Interfaz de usuario para recolectar imágenes de los sensores activados

La interfaz desarrollada no solo permite verificar el funcionamiento de todas las placas de sensores, sino también verificar los archivos JSON generados, ya sea por esta interfaz o por el sistema electrónico que se conecta al laberinto para configurar los experimentos. El sistema electrónico y el formato JSON se explicarán más adelante.

No está de más mencionar que la herramienta DS100 de Azoteq® solo permite leer y configurar un controlador de sensores, debido a que requiere de la señal denominada RDY para validar la lectura de los sensores. Dicha señal es única por cada controlador, por lo que no es posible interconectarlas entre sí, como es el caso de las señales SDA y SCL del protocolo I^2C.

Figura 74: Ejemplo de imagen generada con la interfaz en C++

6.7. Interfaz electrónica de usuario

Los usuarios que realizan experimentos con el laberinto elevado en forma de cruz requieren definir ciertos parámetros antes de usar el sistema automatizado con

sensores capacitivos. Los parámetros que deben seleccionarse en la interfaz del usuario se enlistan a continuación:

- Programar el tiempo de experimentación (mínimo 5 minutos).
- Poner la fecha del experimento.
- Respaldo del experimento.
- Identificación de los distintos experimentos.

Para definir los parámetros, el sistema completo se implementó como se ilustra en el diagrama de la Figura 75.

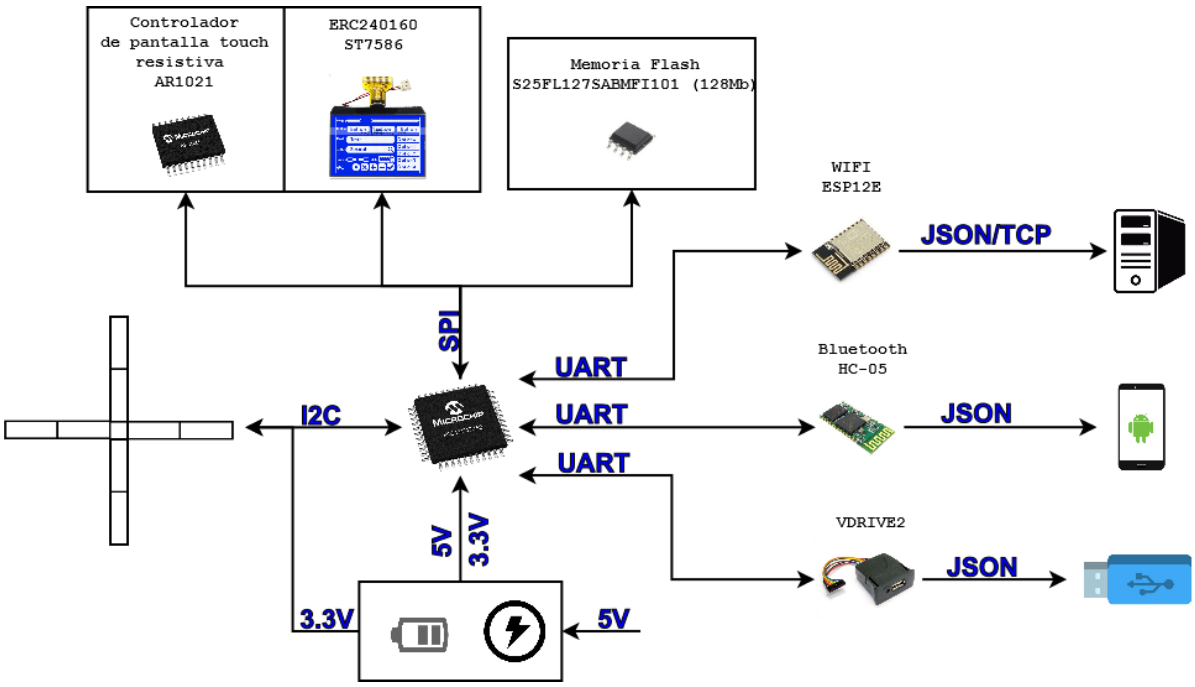


Figura 75: Diagrama de la interfaz electrónica de usuario

El microcontrolador central que se seleccionó fue el PIC18F47J53 de Microchip®, ya que además de ser económico, tiene la cantidad suficiente de memoria RAM para

almacenar variables y una gran cantidad de memoria tipo flash (memoria de programa) para almacenar el código del microcontrolador, todo lo cual es necesario para el almacenamiento de las imágenes que se despliegan en la pantalla táctil. Otra de las características fundamentales que podemos destacar es que cuenta con 2 módulos MSSP, que permiten el control de la comunicación SPI e I^2C por puertos diferentes. También cuenta con 2 módulos UART, uno de los cuales es “remapeable”, es decir, permite seleccionar las líneas de TX y RX en el puerto que se requiera, además de que también, mediante la configuración de registros internos, posibilita el cambio de puerto de estas líneas en tiempo de ejecución, permitiendo la comunicación con distintos dispositivos conectados en diferentes puertos. En la Tabla 10 se pueden observar las características más importantes de este microcontrolador.

Tabla 10: Características principales del microcontrolador central.

PIC18F47J53	
Memoria de programa	Flash 128KB
Memoria RAM	3,800 bytes
Periféricos de comunicación	2-UART, 2-SPI, 2-I^2C
Temporizadores	4x8-bit, 4x16-bit
Voltaje	2 a 3.6 V

El circuito implementado para el microcontrolador se muestra en la Figura 76. En el diagrama también se muestra la conexión con el laberinto y el circuito necesario para la programación del microcontrolador.

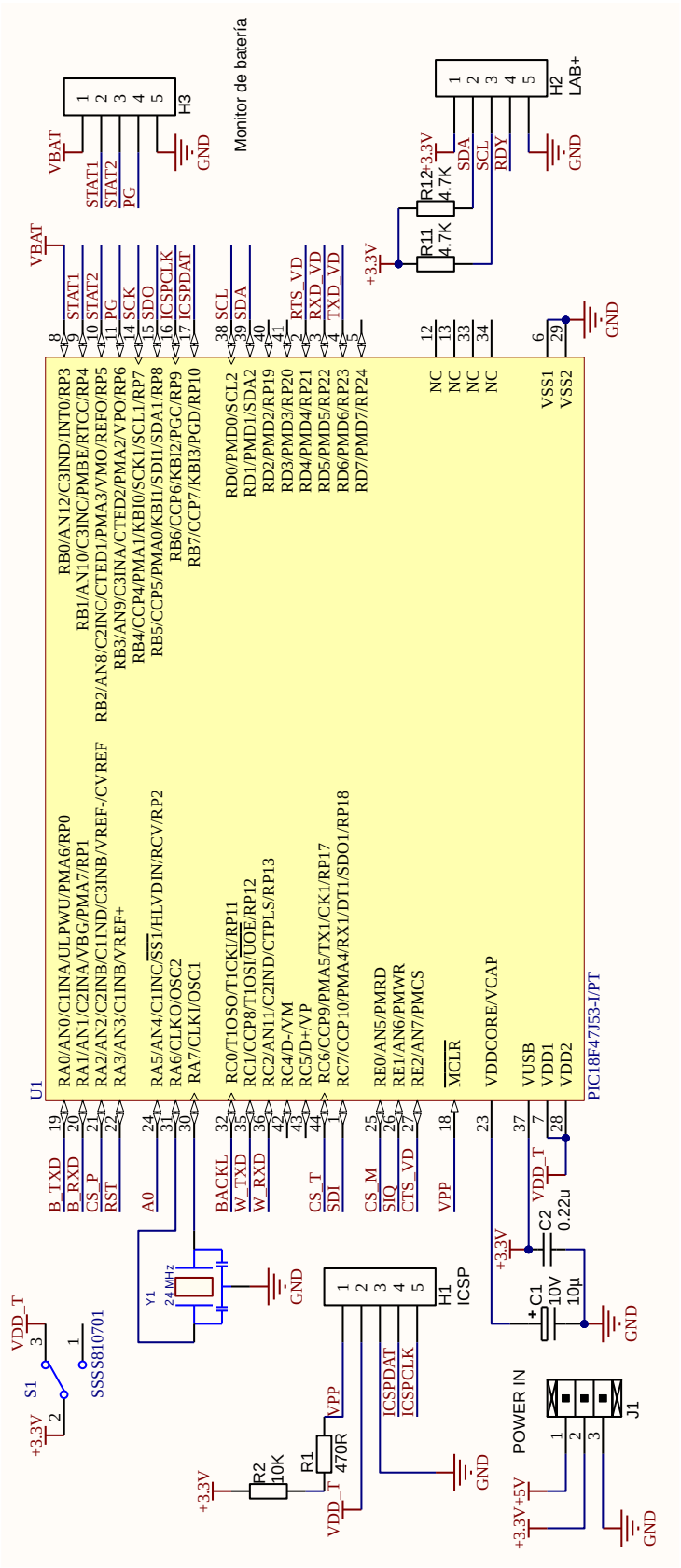


Figura 76: Diagrama del circuito del microcontrolador central

6.7.1. Pantalla táctil

Para lograr una interfaz amigable con el usuario, se eligió en una pantalla táctil dotada de íconos para que el usuario pueda manejar fácilmente el sistema. Para ello se seleccionó la pantalla gráfica ERC240160 que permite el manejo de los íconos. En la Tabla 11 se describen las características de la pantalla, la cual soporta SPI para la comunicación y sus niveles lógicos están basados en 3 V.

Tabla 11: Características de la pantalla

ERC240160	
Fabricante	EastRising
Dimensiones	3.4”
Formato de visualización	240 x 160 puntos
Controlador	ST7586
Interfaz de comunicación	Paralelo de 8 bits, SPI
Luz de fondo	Si, 75 mA
Voltaje	3 V

Para manejar la pantalla mediante el protocolo SPI, se implementó el circuito de la Figura 77.

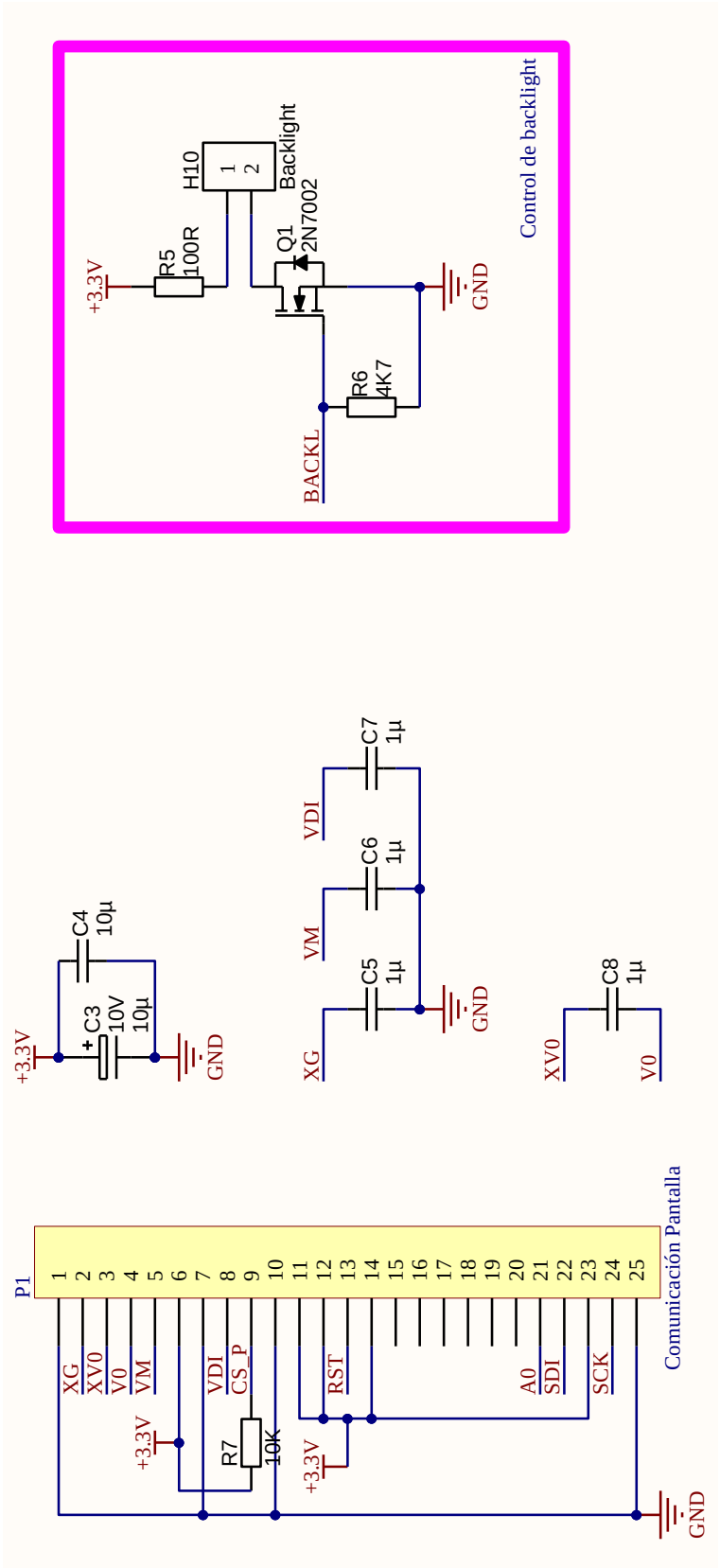


Figura 77: Diagrama esquemático de la pantalla de la interfaz de usuario

Algunos fabricantes de estas pantallas las proveen con un panel táctil resistivo de 4 líneas, el cual requiere de un controlador que permite detectar cuando ha sido tocada la pantalla y en qué posición. Para realizar esta tarea se seleccionó el controlador AR1021 de Microchip®, cuyas características se describen en la Tabla 12.

Para lograr la comunicación con el controlador del panel táctil mediante el protocolo SPI se implementó el circuito de la Figura 78.

Tabla 12: Características del controlador del panel táctil de la pantalla

AR1021	
Resolución	1024 x 1024 pixeles
Líneas táctiles	4, 5, 8
Interfaz de comunicación	I^2C, SPI
Voltaje	2.5 V a 5.5 V

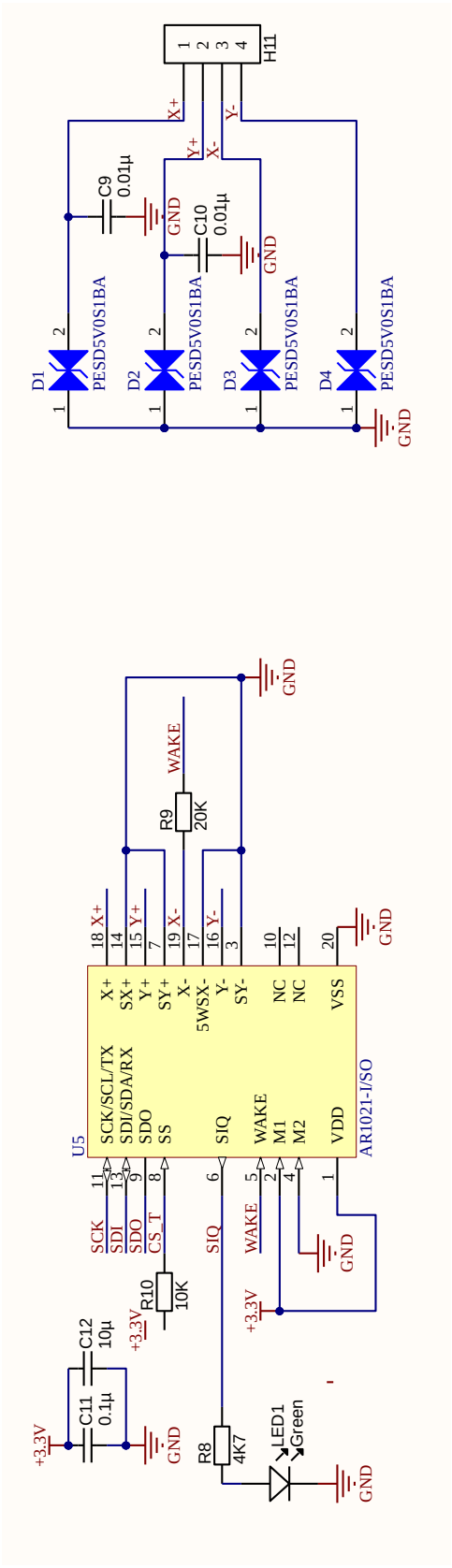


Figura 78: Diagrama del circuito del controlador de la pantalla táctil

La integración de la pantalla táctil proporcionó una interfaz amigable y fácil de usar para el usuario. En la Figura 79 se ilustra la pantalla principal de la interfaz.

Figura 79: Interfaz electrónica de usuario para el laberinto elevado en forma cruz

6.7.2. Respaldo interno de experimentos

El sistema implementado permite respaldar la información de hasta 15 experimentos con duración máxima de 30 minutos cada uno. La información se guarda en una memoria tipo flash S25FL127 de Cypress®, la cual puede almacenar hasta 128 Mbits (16 MBytes) de información. Las principales características de esta memoria son las siguientes:

- Tecnología CMOS 3.0 V.
- Densidad de 128Mbits (16 MBytes o 16,777,216 bytes).
- Interfaz de comunicación SPI en modo 0 y 3.
- Escritura de páginas por búfer de 256 y 512 bytes a una velocidad de 0.8 MBytes/s.

- Borrado por segmentos de 64 KBytes (65,536 bytes) a una velocidad de 0.5 MBytes/s.
- 100,000 ciclos de escritura y borrado.
- Duración de almacenamiento de datos de 20 años en promedio.

El circuito implementado para la memoria se muestra en la Figura 80.

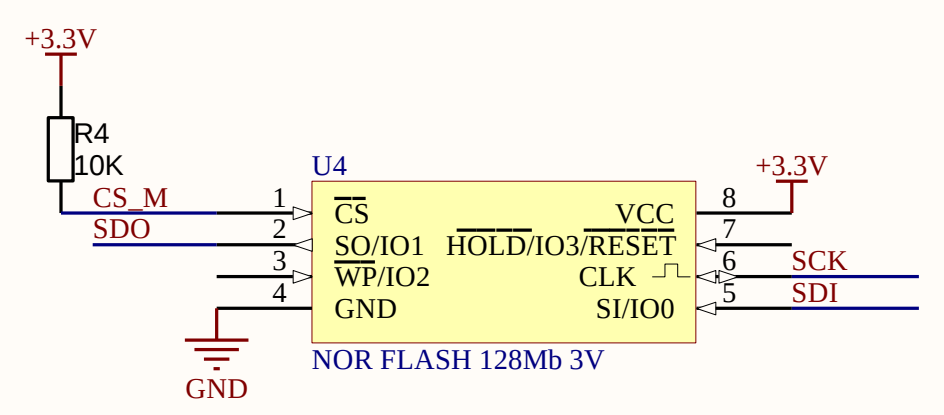


Figura 80: Diagrama del circuito de la memoria flash

Con el fin de organizar la memoria para guardar los experimentos, se consideró como base el tamaño del segmento de borrado de 64 KBytes, ya que al ser memoria tipo flash no permite la escritura en un byte si éste no ha sido borrado previamente, y el borrado se realiza sobre segmentos de 64 KBytes. Por lo tanto, el espacio que ocupa un experimento está en múltiplos de 64 KBytes. Por consiguiente, debido al tamaño del segmento de borrado, los 16 MBytes de la memoria se encuentran divididos en 256 segmentos de 64 KBytes.

Para calcular la cantidad de espacio que ocupa un experimento primero se definió la información que se necesita almacenar y la cantidad de bytes que ocupa. En la Tabla 13 se desglosa la información que se debe guardar en cada experimento, así como su tamaño en bytes. En la Tabla 14 se enlista la información que se guarda de la configuración para una placa de sensores, y en la Tabla 15 la información que se guarda por cada segundo del experimento. Con esta información podemos definir la

siguiente fórmula para calcular el número en bytes que se requiere almacenar para cada experimento.

$$Tam.Exp = Inf.Exp + Conf.Placa \times 9 + Inf.Seg \times Tiempo\ Exp. \times 60 \quad (5)$$

Si consideramos que el tiempo mínimo de experimentación es de 5 minutos, de acuerdo a la ecuación (5) el experimento tendrá un tamaño de 15+[33*9]+[560*5*60] = 168,312 bytes.

Si la duración de los experimentos fuese de 30 minutos, entonces se necesitaría almacenar 1,008,312 bytes (aprox. 1MByte). Considerando que la memoria está dividida en segmentos de 65536 bytes, entonces cada experimento de 30 minutos requeriría de 1,008,312 / 65536 = 15.38 segmentos, que son en realidad 16 segmentos. Lo anterior permite almacenar hasta 256/16 = 16 experimentos. Sin embargo, el número máximo de experimentos de 30 minutos que se pueden almacenar se fijó a 15 y el resto de los segmentos se utilizó para almacenar información de la configuración del sistema, tales como el nombre de la red WiFi, contraseña, la última configuración de experimento usada, etc.

Tabla 13: Información del experimento y tamaño que ocupa en la memoria

Información	Tamaño en bytes
Fecha (d/m/a)	3
Identificador de rata (2 caracteres y un número)	3
Tiempo de experimentación (minutos)	1
Orden en el que están organizadas las placas de sensores	8

Total	15

Tabla 14: Configuración de una placa de sensores y su tamaño que ocupa en la memoria

Información	Tamaño en bytes
Identificador de placa de sensores	1
Configuración de canales activos	30
Total de Rx	1
Total de Tx	1
Total	33

Tabla 15: Información guardada por segundo y su tamaño que ocupa en la memoria

Información	Tamaño en bytes
Segundo	2
Identificador de placa de sensores	1
Valores de proximidad	30
Valores de toque	30
Número de dedos detectados	1
Total	2 + [(1 + 30 + 30 + 1) x 9] = 560

6.7.3. Respaldo externo de experimentos

El respaldo externo de los experimentos consiste en la lectura de la información almacenada en la memoria tipo flash y su transferencia por distintos medios externos: Bluetooth, WiFi o memoria USB. Para realizar la transferencia, a la información leída de la memoria se le dio un formato sencillo de comprender para los usuarios y a la vez práctico para leer y procesar mediante cualquier lenguaje de programación. El formato que se eligió fue JSON, ya que es uno de los formatos más usados en la actualidad para intercambio de información.

El formato de la configuración del experimento se definió como el ejemplo que se muestra en la Figura 81. El campo “device” indica el dispositivo de experimentación, que en este caso es el laberinto elevado en forma de cruz, el cual está representado por la cadena de texto “EPM” que son las siglas de su nombre en inglés “Elevated Plus Maze”. La fecha se encuentra en el campo “date”. El campo “id” es el identificador del animal en experimentación. El campo “texp” indica el tiempo que dura el experimento, que en el ejemplo es de 1 minuto. Por último, el campo “arrange” es el arreglo de las posiciones de las placas de los brazos en el laberinto elevado en forma de cruz.

```
"device": "EPM",
"date": "23/03/2017",
"id": "EV01",
"texp": 1,
"arrange": [
  1,
  2,
  3,
  4,
  7,
  8,
  5,
  6
],
```

Figura 81: Ejemplo JSON de configuración de experimento

Para guardar la configuración de cada placa, el formato JSON se definió como se muestra en el ejemplo de la Figura 82. El campo “config” indica el arreglo de

configuraciones de las 9 placas, donde cada elemento del arreglo es la configuración de una placa de sensores. El campo “id” representa el identificador de la placa y el campo “actives”, representa la configuración de los canales que están habilitados en la matriz de sensores.

El formato de la información que se recolecta cada segundo de los sensores se definió como el ejemplo de la Figura 83. El campo “data” indica el arreglo de los datos generados por segundo; por ejemplo, si el experimento dura 1 minuto, entonces “data” contendrá 60 elementos. El campo “second” indica el segundo en el que se recolectó su respectiva información. El campo “values” indica el arreglo de datos por placa; como se cuentan conectadas 9 placas, el tamaño del arreglo es de 9 elementos. El campo “id” es el identificador de la placa. El campo “fingers” indica la cantidad de objetos detectados sobre la placa respectiva. Los campos “prox” y “touch” son arreglos de 30 elementos que indican los valores proximidad y toque respectivamente.

```
"config": [
    {
      "id": 0,
      "actives": [
        3,
        190,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        190,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        255,
        3,
        255
      ],
      "totalrx": 7,
      "totaltx": 6
    },
```


Figura 82: Ejemplo JSON de la configuración de una placa de sensores

```
"data": [
    {
      "second": 0,
      "values": [
        {
          "id": 0,
          "fingers": 2,
          "prox": [
            0,
            0,
            0,
            64,
            0,
            64,
            0,
            0,
             .
             .
             .
             (30 elementos)
          ],
          "touch": [
            0,
            0,
            0,
            64,
            0,
            64,
            0,
            0,
             .
             .
             .
             (30 elementos)
          ]
        },
```


Figura 83: Ejemplo JSON de la información recolectada por segundo.

Mediante la interfaz táctil el usuario puede realizar la transferencia de información por cualquiera de los tres medios mencionados anteriormente.

Para la transferencia de los datos experimentales por medio de Bluetooth se utilizó el firmware y la aplicación para el sistema operativo Android, desarrollados en el laboratorio (Chacón Matú, 2016). La transferencia por este medio se realizó utilizando el módulo HC-05. El diagrama del circuito para el módulo se muestra en la Figura 84.

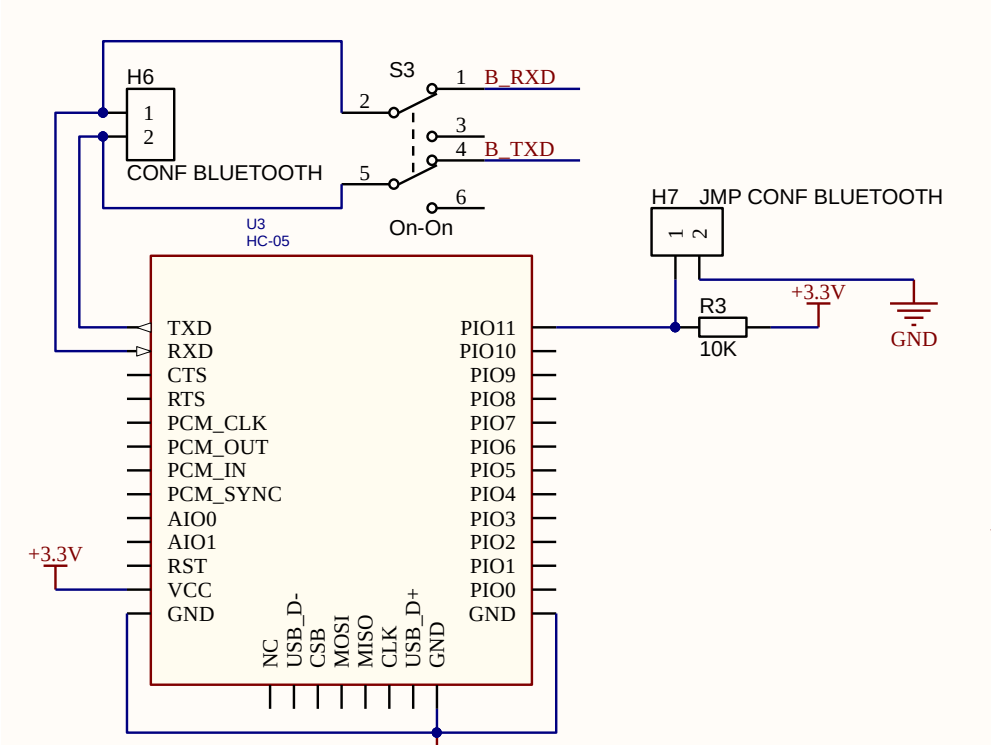


Figura 84: Diagrama del circuito para el módulo Bluetooth HC-05

Para la transferencia del archivo a la memoria USB se migró el firmware ya desarrollado con anterioridad en el laboratorio (Collí Alfaro, 2014). El dispositivo que realiza esta tarea es el VDRIVE2, cuya conexión se realizó como se muestra en la Figura 85.

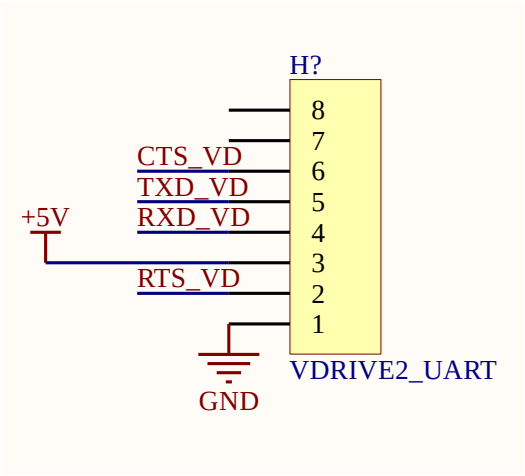


Figura 85: Diagrama del circuito para la conexión del VDRIVE2

El laboratorio cuenta con un servidor web para la administración de experimentos por usuarios (Chacón Matú, 2016). Para poder insertar experimentos en la base de datos el servidor cuenta con un script desarrollado en el lenguaje Python, encargado de monitorear solicitudes en un puerto específico por medio del protocolo TCP/IP.

La transferencia por medio de WiFi permite el almacenamiento de información a la base de datos del servidor. Para ello se implementó el firmware para enviar el archivo por medio del protocolo TCP/IP.

El módulo WiFi utilizado para llevar a cabo la conexión con el servidor fue el ESP-12E. La comunicación entre el microcontrolador y el módulo se realiza mediante el puerto UART. El módulo WiFi requiere que se le envíen comandos estándar denominados “ATCommands”. El circuito necesario para la comunicación con el módulo WiFi se muestra en la Figura 86.

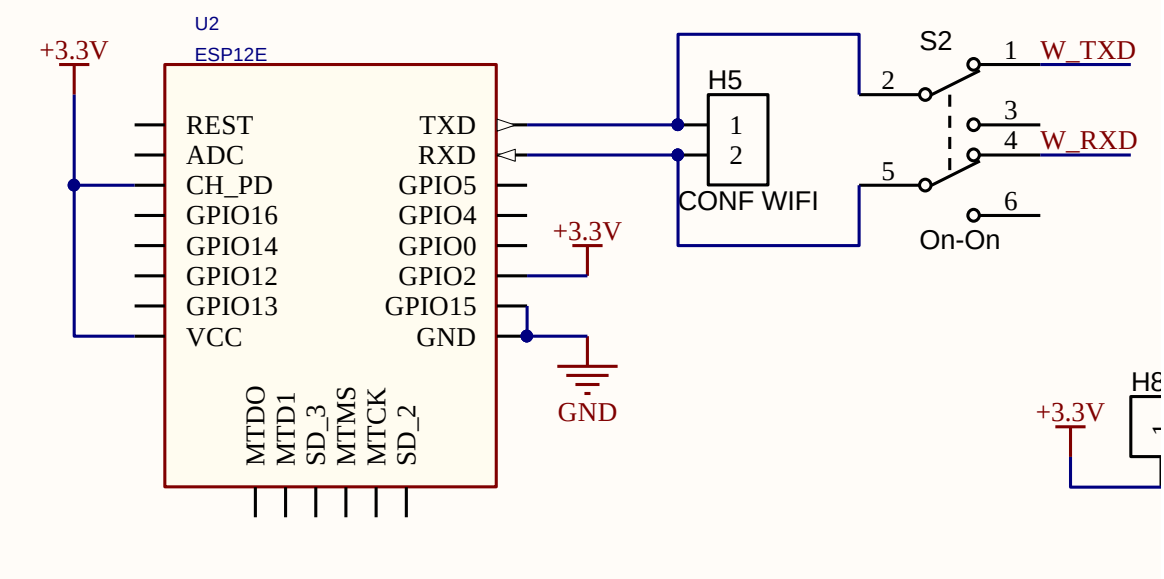


Figura 86: Diagrama del circuito para el módulo ESP-12E

6.7.4. Fuente y respaldo de energía

Para el diseño de la fuente que provee de energía al sistema se consideró el mayor consumo posible de corriente de los dispositivos. Para ello se consideró un consumo de 500 mA para la fuente de 5 V cuando se conecta la memoria USB al VDRIVE2 y de 500 mA para la fuente de 3.3 V utilizada por el resto de los dispositivos, dando como resultado un total de 1 A de consumo máximo.

La fuente se diseñó para operar a partir de una fuente externa de 5 V o por medio de una batería de Ion Litio de 3500 mAh con un voltaje nominal de 3.7 V, capaz de durar hasta 3.5 horas en caso de un corte de energía, en el peor de los casos.

Para el circuito de carga de la batería se optó por utilizar el circuito integrado BQ24266 de Texas Instruments® el cual puede generar una corriente de carga de hasta 3 A y soporta un voltaje máximo de entrada de 30 V. Se eligió este dispositivo debido a que es capaz de dividir su fuente de entrada para alimentar al sistema y cargar a la batería al mismo tiempo. Cuando ocurre un corte de energía, o no hay voltaje de entrada, el dispositivo automáticamente habilita la batería como fuente de energía para el sistema. El cargador monitorea la corriente de la batería todo el tiempo y reduce su corriente de

carga cuando el sistema requiere mayor corriente que la corriente de entrada programada, o si la fuente de entrada no provee suficiente corriente al sistema. La arquitectura del dispositivo también permite a la batería proveer de corriente al sistema cuando el voltaje de entrada no soporta los picos de corriente del sistema. En la Figura 87 se muestra una representación del manejo de voltaje para el sistema.

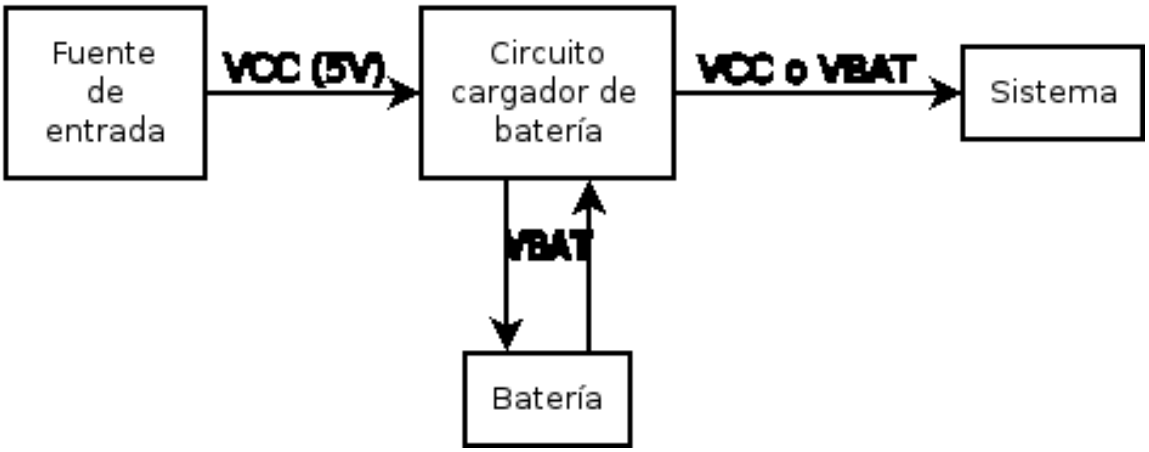


Figura 87: Diagrama de bloques del sistema de carga

Basándose en la información referente al dispositivo de carga y a los requerimientos de energía se diseñó el circuito que se muestra en la Figura 88.

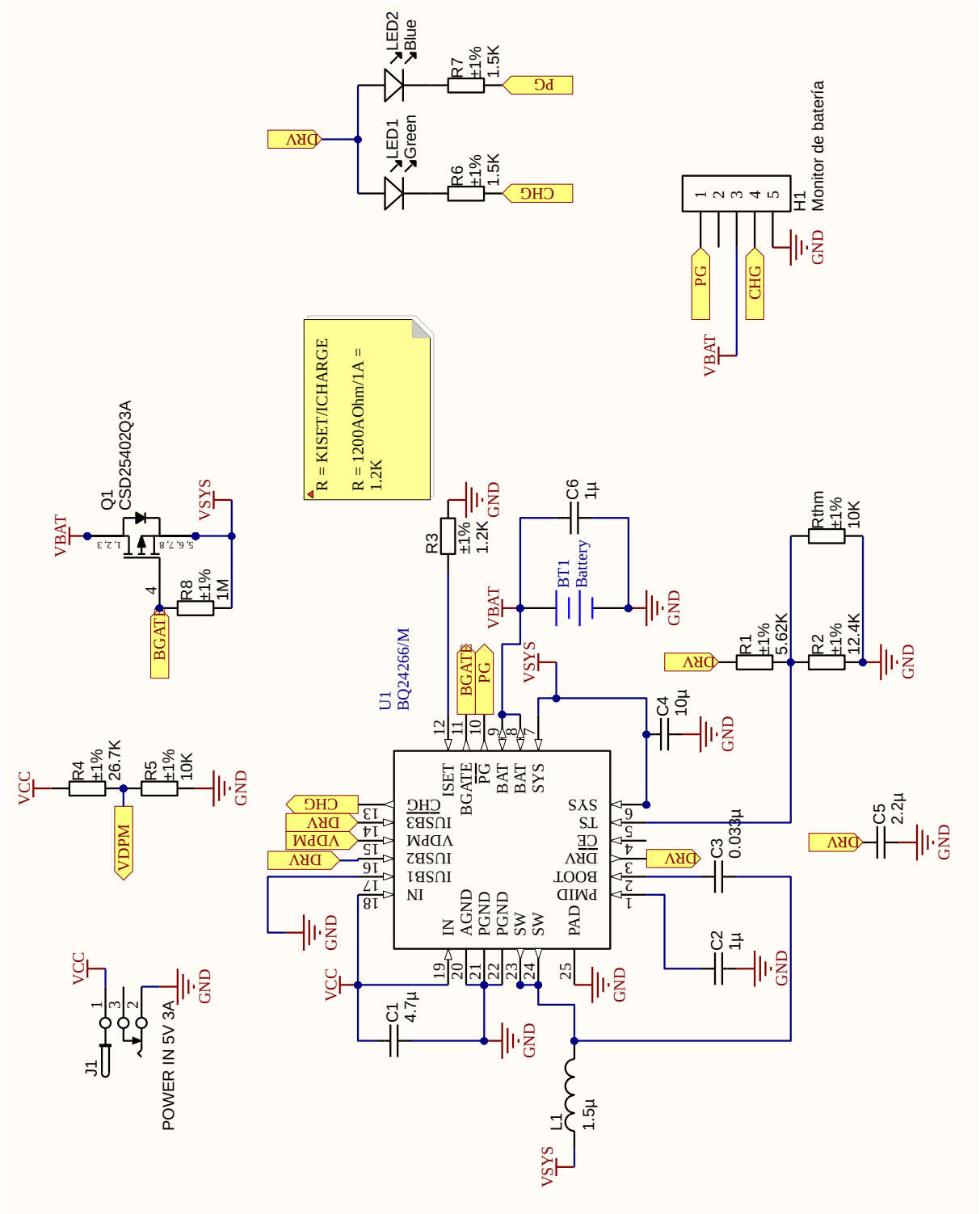


Figura 88: Diagrama del circuito del cargador de baterías

La señal DRV es un voltaje generado por un regulador lineal interno del circuito cargador que se utiliza para polarizar a los transistores MOSFETs internos, aunque también puede utilizarse para manejar señales externas de hasta un máximo de 10 mA. El regulador solo está activo cuando existe una fuente externa de voltaje conectada.

El circuito cargador puede trabajar de diferentes maneras de acuerdo a la configuración de sus señales de entrada IUSB1, IUSB2 e IUSB3. En este caso se configuró para limitar la corriente de entrada a 2.5 A, por lo que las señales quedaron como sigue: IUSB1 = 0, IUSB = 1 e IUSB3 = 1.

Para determinar la corriente de carga de la batería, se requiere colocar una resistencia (R3) en la señal de entrada ISET del circuito. Para realizar el cálculo se utilizó la fórmula que se muestra en la ecuación (6). Donde K_{ISET} es un parámetro del circuito integrado con el valor constante de 1200 AOhm, e I_{CHARGE} es la corriente de carga para la batería, la cual se fijó a 1000 mA. Realizando la sustitución de los valores en la ecuación obtenemos que R3 = 1.2 kOhm.

$$R3 = K_{ISET}/I_{CHARGE} \quad (6)$$

Con el diseño de la batería tenemos que la magnitud del voltaje que alimenta al sistema está en un rango que va desde 2.8 V, valor mínimo de la batería, hasta un máximo de 5 V, debido a la fuente de alimentación. Sin embargo, los circuitos del sistema requieren una fuente constante de 5 V y otra constante de 3.3 V para su correcto funcionamiento.

Para lograr un voltaje constante de 5 V a partir del intervalo de voltajes antes mencionados fue necesario incluir un convertidor DC – DC tipo boost para poder incrementar y fijar el voltaje de entrada a 5 V. Para ello se utilizó el circuito integrado TPS61085T de Texas Instruments®, que es capaz de convertir un voltaje de entrada de un rango de 2.3 a 6 V, hasta a un voltaje igual a su entrada más 0.5 V, o hasta 18.5

V con una corriente conmutada de 2 A y una frecuencia de 650 kHz o 1.2 MHz. El circuito diseñado se muestra en la Figura 89.

Para fijar el voltaje de salida a 5 V se requiere colocar el valor de un voltaje en la señal de FB (Feedback o retroalimentación). Para ello, lo usual es utilizar un circuito divisor de voltaje mediante resistencias (R10 y R11). Por recomendaciones del fabricante el valor de la resistencia R11 se fijó a 18 kOhm con el fin de lograr una corriente de retroalimentación de al menos 50 µA, y la resistencia R10 se calculó empleando la ecuación (7), donde Vs es el voltaje de salida deseado (5 V) y Vref es un parámetro constante de 1.24 V, dando como resultado un valor de R10 de aproximadamente 56 kOhm.

$$R10 = R11 \times \left(\frac{Vs}{Vref} - 1\right) \tag{7}$$

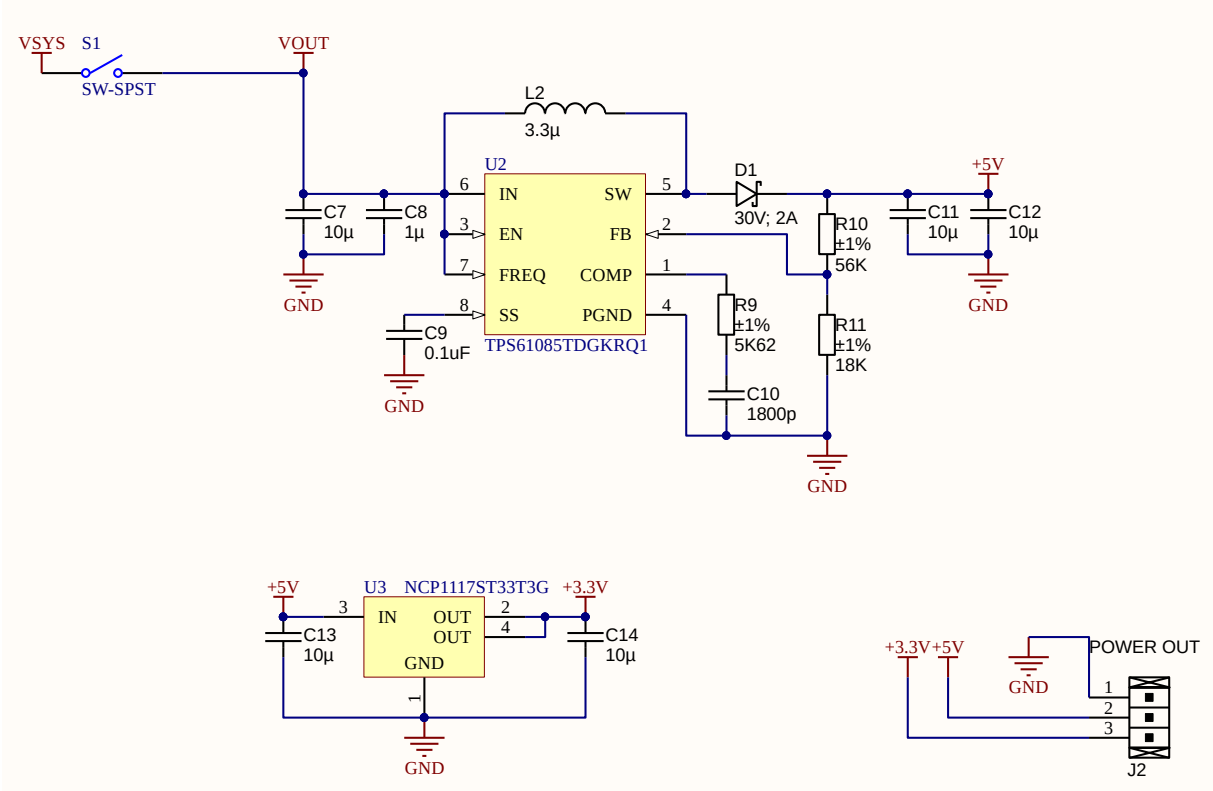


Figura 89: Diagrama de la fuente de voltaje de 5 V y 3.3 V

Los valores del resto de los componentes son los recomendados por el fabricante en la hoja de datos del dispositivo.

Para la fuente fija de 3.3 V se utilizó el regulador lineal NCP1117ST33 del fabricante ON Semiconductor®. Como fuente de entrada se utilizó la salida del convertidor DC – DC. El circuito puede observarse en la Figura 89.

Para garantizar que la batería sea cargada mientras el sistema está apagado, se colocó un interruptor entre la salida del cargador de baterías y el convertidor DC – DC.

6.8. Algoritmo de detección de la rata

Para determinar la ubicación de la rata en el laberinto se implementó un algoritmo en el lenguaje de programación gráfica LabVIEW. Se eligió este lenguaje debido a que el CIR-UADY ya cuenta con un software en el ambiente LabVIEW diseñado para procesar datos de distintos laberintos, el cuál puede usarse también para el procesamiento de datos del laberinto elevado en forma de cruz.

Lo primero que se realizó fue la lectura e interpretación del archivo en formato JSON. Para ello, se instalaron bibliotecas existentes para obtener los valores de los campos del archivo JSON. La biblioteca que se instaló se denomina JSON API y fue desarrollada por la empresa LAVA. Esta biblioteca cuenta con un módulo que convierte un texto en formato JSON a un tipo de dato en el que se pueden obtener los valores de los campos. En la Figura 90 se muestra cómo se realiza la lectura de la configuración del experimento utilizando estas bibliotecas.

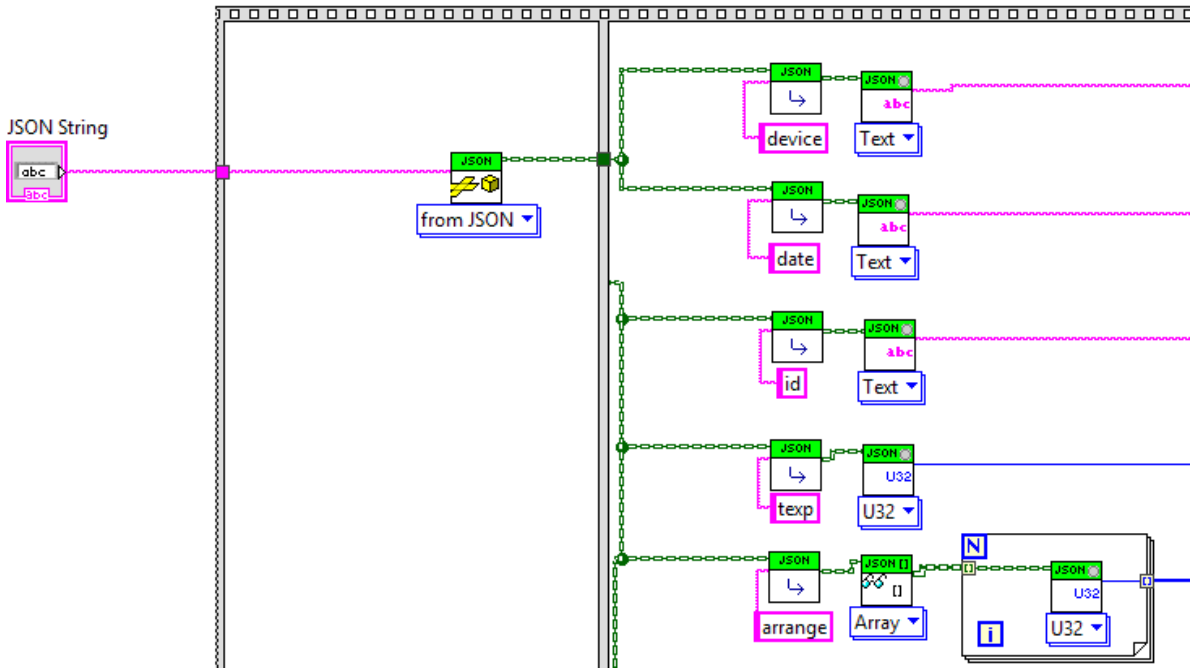


Figura 90: Lectura de la configuración del experimento en LabVIEW

Para la lectura de las configuraciones de las 9 placas de sensores se realizó mediante un ciclo “for”. En la Figura 91 se observa cómo se realizó la implementación y también se puede notar que se incluyó un módulo para construir la matriz de los canales activados mediante la lectura del arreglo leído del JSON y el valor de totalrx y totaltx.

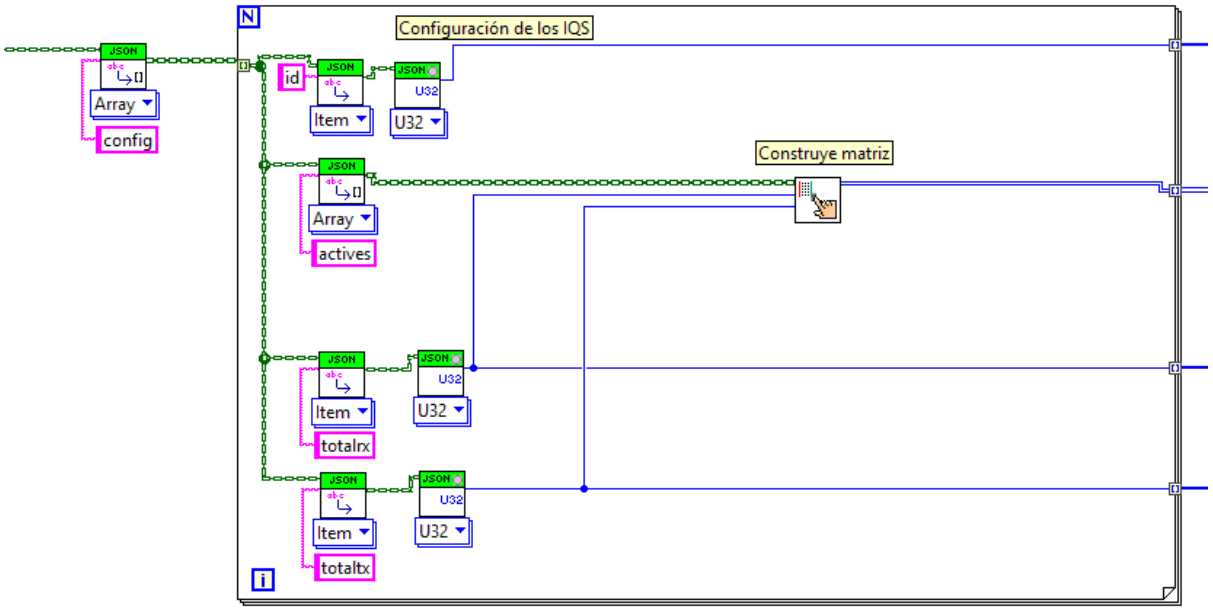


Figura 91: Software de lectura de la configuración de las 9 placas de sensores capacitivos en LabVIEW

De la misma manera se realizó la lectura de los valores de toque y proximidad de las 9 placas por segundo (Figura 92).

El módulo que construye la matriz de canales activos, toque y proximidad a partir de los valores de totalrx y totaltx se implementó como se muestra en la Figura 93.

La interpretación del texto en formato JSON se integró en un solo módulo de LabVIEW (Figura 94), permitiendo así el manejo numérico de los datos para el algoritmo que detecta la posición del roedor en el laberinto.

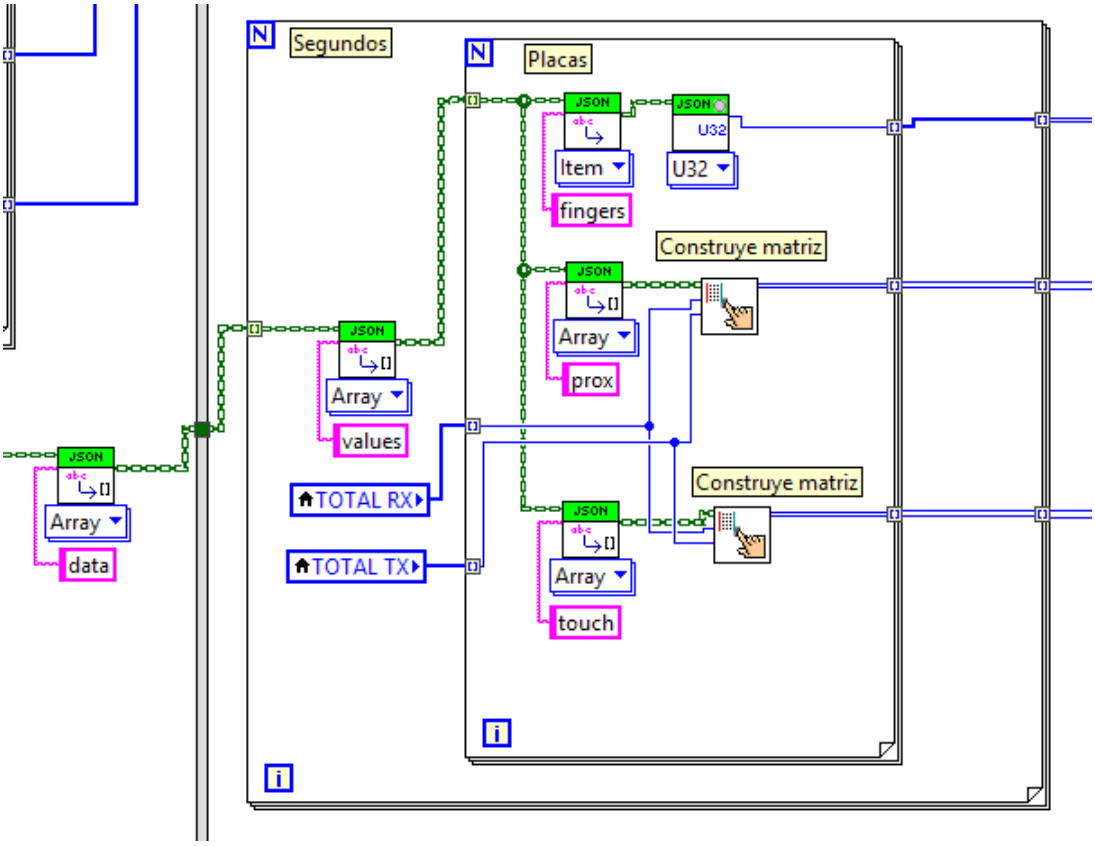


Figura 92: Software de lectura de los valores de toque y proximidad en el lenguaje LabVIEW

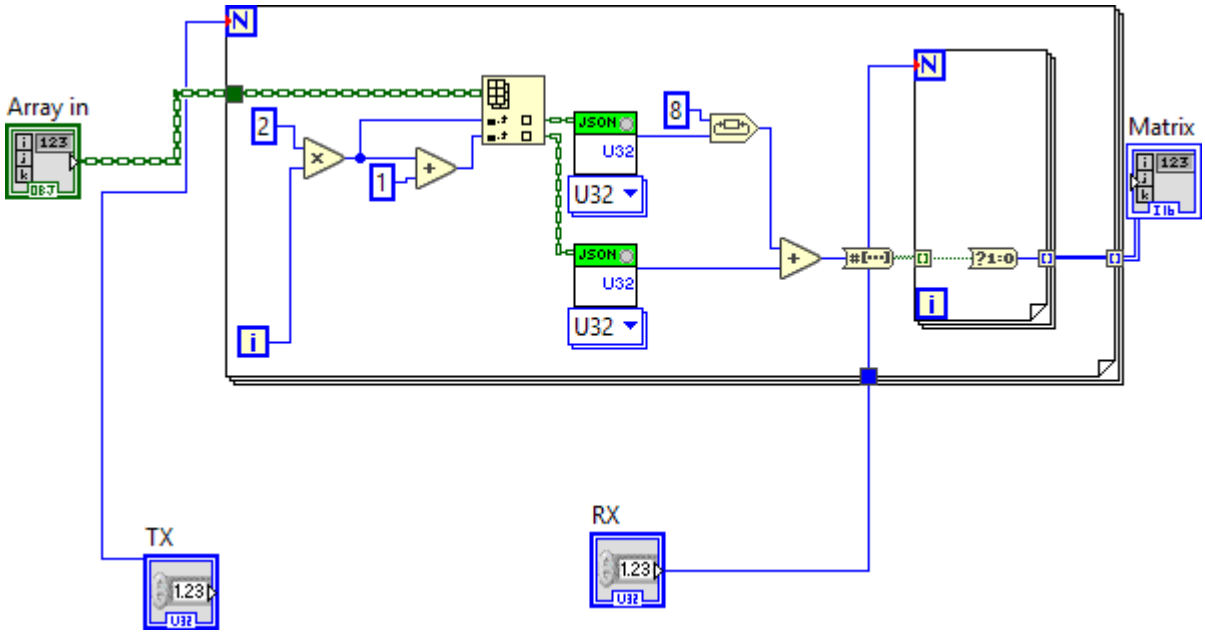


Figura 93: Módulo para la construcción de una matriz en LabVIEW

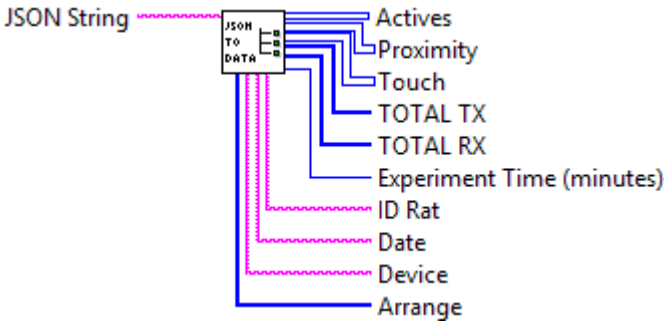


Figura 94: Módulo para la interpretación del texto en formato JSON

Para la realización del algoritmo, primero se realizaron pruebas para obtener datos y posteriormente observar el comportamiento de los mismos. Las pruebas que se realizaron se detallarán en el apartado de DISEÑO EXPERIMENTAL.

De cada placa se obtuvieron 3 fuentes de información: estados de proximidad, de toque y número de dedos en la placa. Para el algoritmo, simplemente se consideraron los estados de proximidad y de toque. Sin embargo, debido a problemas de comunicación no se pudieron obtener de manera fiable los datos, lo cual se explicará en el apartado de RESULTADOS.

El estado de proximidad y de toque están representadas en una matriz de 1's y 0's, es decir, que cada canal o sensor tiene un 1 si alguna parte del cuerpo de la rata está ocupando el sensor, en caso contrario está en 0 si no hay nada en el sensor. La diferencia entre la proximidad y el toque es que el sensor es más sensible al valor de proximidad que al valor de toque. Por lo anterior se pensó en crear un algoritmo simple para contar el número de sensores en estados de proximidad y de toque. Sin embargo, por ser menos sensible, el valor de toque fue multiplicado por 2, de tal forma que si un sensor es activado por toque es indicativo de una mayor probabilidad de que la rata esté en ese sensor. Por consiguiente, si una placa tiene muchos sensores en estado de toque, es más probable que la mayor parte del cuerpo se encuentre ocupando esa placa que en otra que tiene menos sensores en toque. De manera similar se mantienen en balance la cantidad de sensores en estado de proximidad y de toque, ya que por su mayor sensibilidad usualmente se encontrará un mayor número de sensores en estado de proximidad que en estado de toque. Al realizar esta operación obtenemos

un valor por cada placa, como se muestra en la ecuación (8), en donde SUM realiza la suma de los valores de una matriz. Como resultado de realizar esta operación para cada placa, obtenemos un arreglo de 9 valores por cada segundo del experimento.

$$V_i = 2 \times SUM(M_{toque}) + SUM(M_{proximidad}) \quad (8)$$

Con el vector denominado “arrange” que se obtuvo del JSON es posible determinar la ubicación de cada placa en el laberinto, permitiendo de esta manera unir los valores de las placas que están en el mismo brazo, ya que representan una misma zona. Al unir dos placas que se encuentran en el mismo brazo se realiza la suma de sus valores calculados para el algoritmo. Al realizar esta operación obtenemos un arreglo de 5 elementos, en donde cada uno representa a uno de los cuatro brazos del laberinto o a la zona central.

Para determinar en qué zona se encuentra el roedor, se localiza el elemento del arreglo que tenga el valor más alto, indicativo de que tiene la mayor cantidad de sensores habilitados. De esta forma se obtendrá un número del 0 al 4 por cada segundo de experimentación, el cual indica en que zona se encuentra el roedor. Contando la ocurrencia de cada uno de estos números podemos saber la cantidad de segundos que la rata estuvo en cada zona. Por último se realiza la suma de las ocurrencias de los dos brazos abiertos y de los dos brazos cerrados, obteniendo de esta manera la cantidad de tiempo en segundos que la rata estuvo en la zona central, los brazos abiertos y los brazos cerrados.

La implementación del algoritmo en el lenguaje de LabVIEW se muestra en la Figura 95.

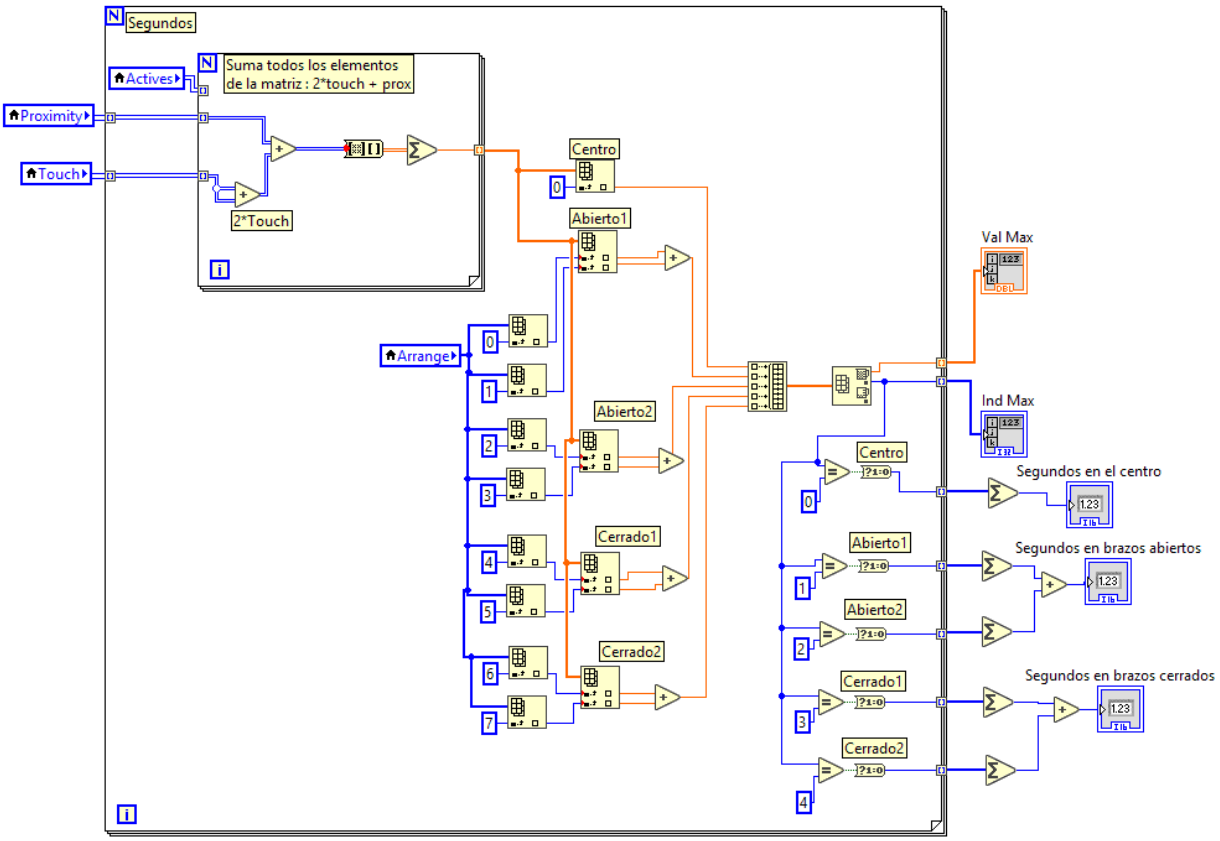


Figura 95: Implementación del algoritmo de detección del roedor en el laberinto

7. DISEÑO EXPERIMENTAL

El diseño experimental original era realizar pruebas con ratas Wistar machos, adultas, sometidas a tratamientos farmacológicos que aumentan o disminuyen la ansiedad. Sin embargo, debido a las limitantes encontradas en la topología de comunicación para la recolección de datos, para evaluar el sistema desarrollado con las características propuestas al inicio se decidió realizar un solo experimento con una rata, sin someterla a algún tratamiento farmacológico. Para ello se programó al sistema para que recolecte la información de los sensores durante 5 minutos y la rata fue colocada en el centro del laberinto elevado en forma de cruz.

Simultáneamente se inició la filmación con una videocámara colocada arriba del laberinto con el fin de tener una vista completa de toda la superficie del laberinto y de

la posición de la rata durante su caminata en las diferentes zonas. Posteriormente se realizó una comparación del resultado del análisis automático de los datos obtenidos de los sensores capacitivos con el análisis del video realizado por observadores, quienes contabilizaron manualmente las entradas y los tiempos de permanencia de la rata en los brazos abiertos, los brazos cerrados y la zona central del laberinto.

En la Figura 96 se observa el cuarto de experimentación donde se realizó la prueba.

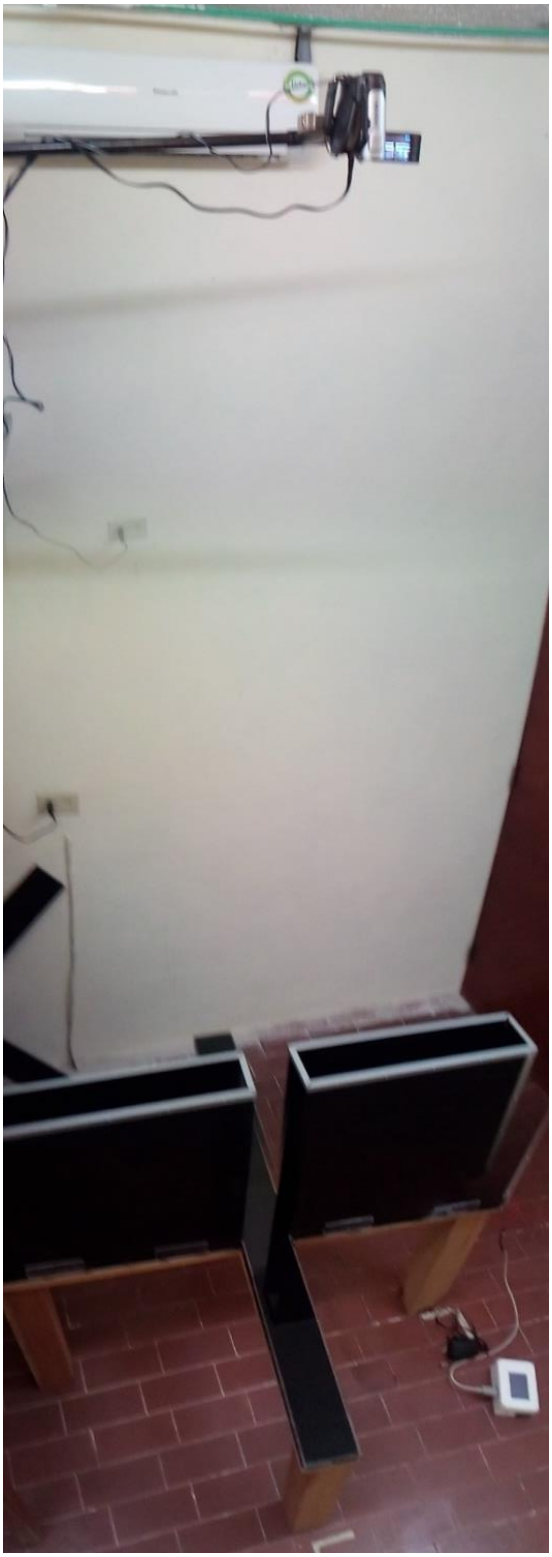


Figura 96: Cuarto de experimentación

8. RESULTADOS

Para comparar visualmente los resultados, los datos obtenidos del laberinto fueron convertidos a imágenes utilizando la herramienta desarrollada en C++ explicada en el apartado 6.6. Aunque en algunas ocasiones la comunicación funcionó adecuadamente, en muchos casos el sistema fue incapaz de ubicar la posición de la rata.

En la Figura 97 se ilustra un ejemplo de comunicación adecuada por el sistema automatizado, donde la imagen de la posición de la rata detectada por los sensores tiene buena correspondencia con la imagen de la posición de la rata capturada por el video en el mismo segundo. Sin embargo, en muchas ocasiones la activación de los sensores no coincidió con la posición real de la rata (Figura 98) y en otros casos los sensores no se activaron (Figura 99).

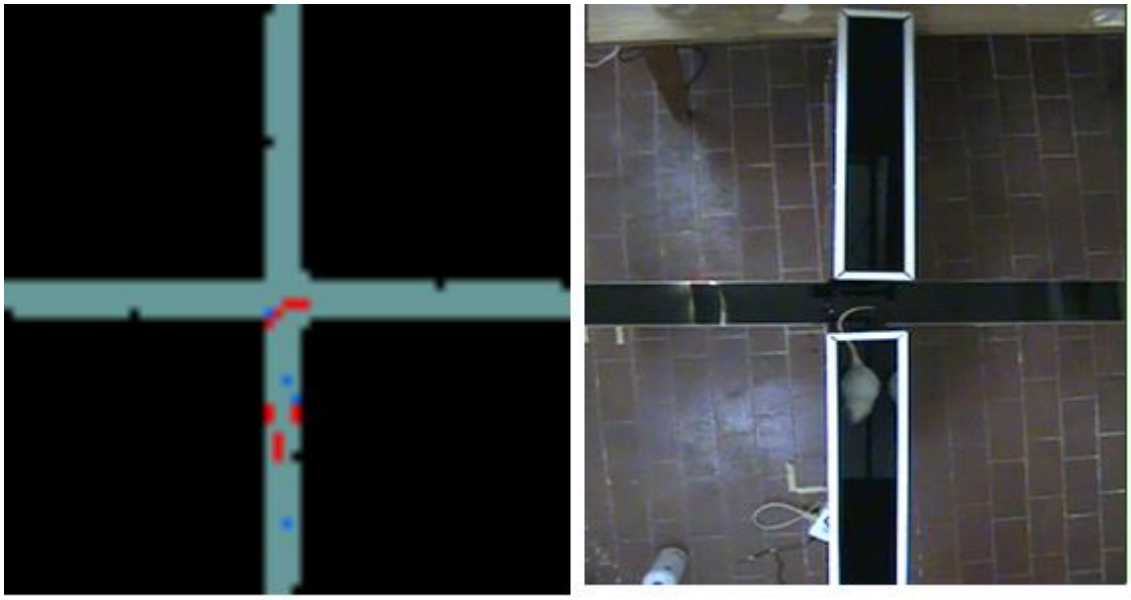


Figura 97: Concordancia entre (A) la activación de los sensores capacitivos (azul proximidad y rojo toque) y (B) la ubicación de la rata en el laberinto capturada en video

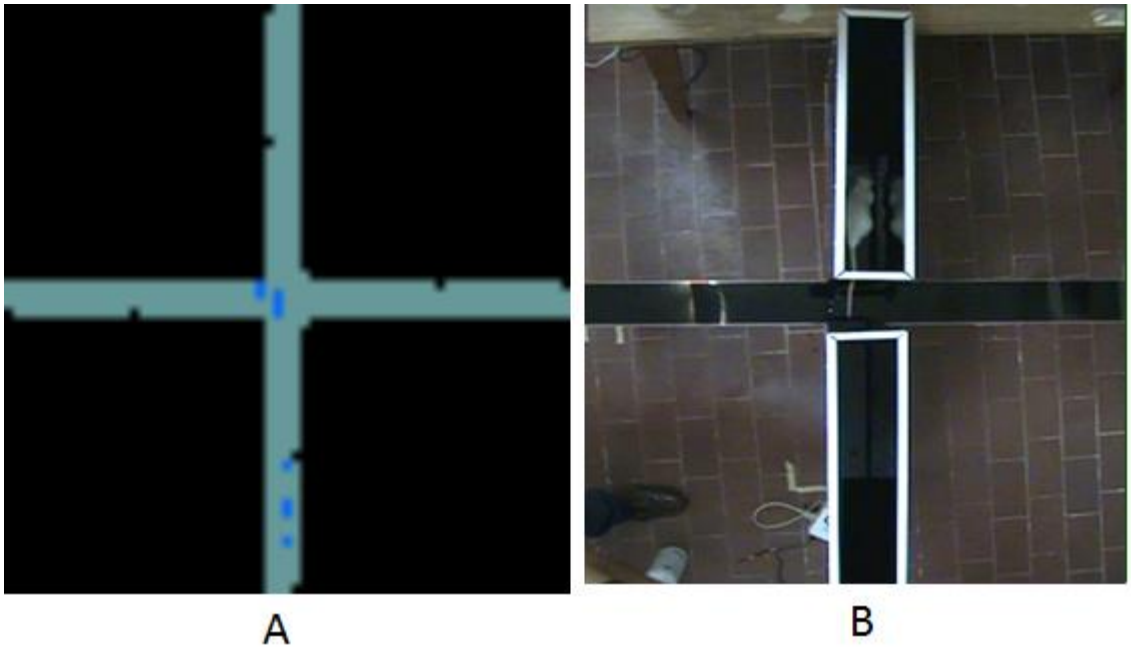


Figura 98: Discordancia entre (A) la activación de los sensores capacitivos (azul proximidad) y B) la ubicación de la rata en el laberinto capturada en video

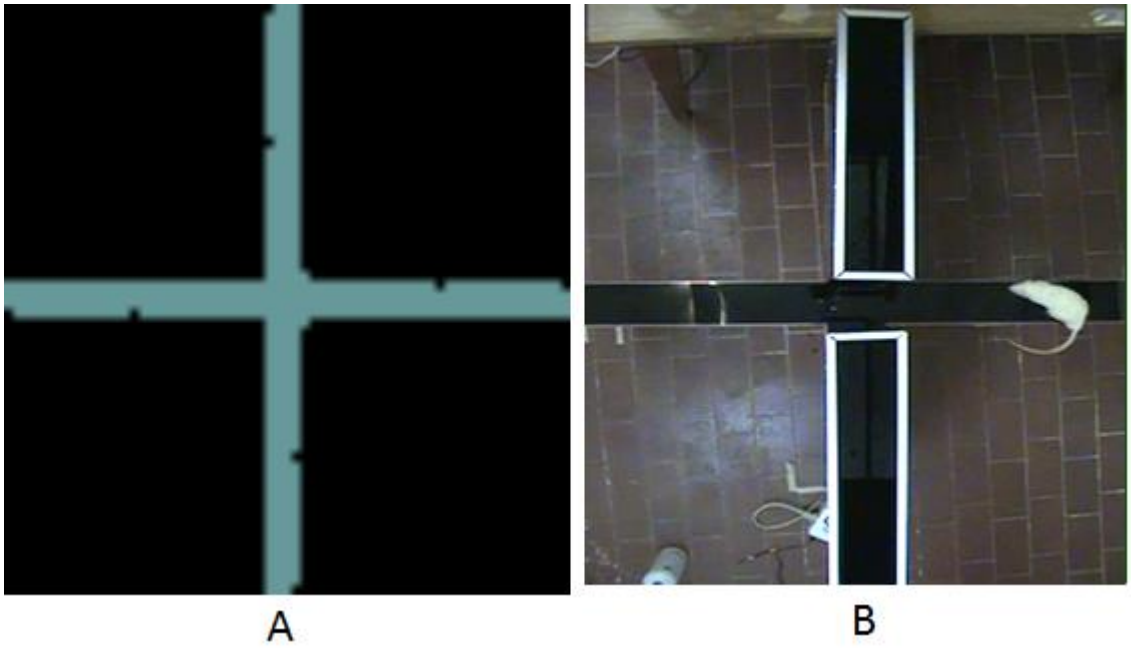


Figura 99: Comparación de (A) la falta de activación de los sensores capacitivos con (B) la ubicación de la rata en el laberinto capturada en video

El resultado obtenido del análisis manual y el obtenido al procesar los datos de los sensores se muestra en la Tabla 16.

Tabla 16: Comparación de los tiempos de permanencia en las zonas del laberinto en forma de cruz contabilizados con el método de observación vs. el automatizado

Zona	Manual (segundos)	Automático (segundos)
Brazo Abierto 1	30	25
Brazo Abierto 2	30	23
Brazo Cerrado 1	97	79
Brazo Cerrado 2	62	91
Centro	81	82
Total Brazos Abiertos	60	48
Total Brazos cerrados	159	170

9. CONCLUSIONES

Aunque el sistema con sensores capacitivos fue capaz de detectar correctamente la presencia de la rata en varias partes del laberinto, en muchas ocasiones los sensores no fueron activados por la presencia del roedor, o se activaron cuando no estaba presente en esa zona. Lo anterior ocurrió por un problema de comunicación ocasionado por la lentitud en la transferencia de los datos que impidió capturar con fidelidad la posición de la rata. Puesto que el laberinto fue dividido en 9 zonas sensoras, el microcontrolador no fue capaz de leer simultáneamente la información de todas las zonas debido a que el protocolo de comunicación usado para la lectura de éstas (I^2C) se realiza de manera secuencial, es decir, que se lee la información de una placa a la vez en forma sucesiva, hasta leer la información de todas. Dada la gran

cantidad de información leída y transferida este proceso se vuelve muy lento en comparación con la velocidad de desplazamiento de la rata, de forma tal que en muchas ocasiones el sistema es incapaz de ubicar su posición real.

Sin embargo, aunque el algoritmo que se implementó es demasiado simple, en varias ocasiones si se realizó exitosamente la captura de la información de los sensores. De hecho, como pudo observarse en algunas imágenes, los sensores activados forman el contorno del cuerpo del roedor. Lo anterior indica que, si implementa un algoritmo más sofisticado para resolver el problema de comunicación con los sensores, si será posible detectar segundo a segundo la ubicación de la rata en todas las zonas del laberinto elevado en forma de cruz.

10. TRABAJO FUTURO

Como trabajo futuro se propone la corrección del sistema de comunicación con los sensores y la mejora del algoritmo de detección de la rata. Para la corrección de la comunicación se planea adicionar a cada matriz de sensores capacitivos un microcontrolador y una memoria tipo flash dedicados a la tarea exclusiva de recolectar y almacenar la información generada en cada placa. Este proceso dará inicio cuando el microcontrolador central indique que ya comenzó el experimento, permitiendo la recolección simultánea de información en todas las placas de sensores, evitando la pérdida de información para que el sistema ubique correctamente la posición del animal en cada instante del experimento.

11. BIBLIOGRAFÍA

Akhtar, H., Kakarala, R., & Member, S. (2014). A Methodology for Evaluating Accuracy of Capacitive Touch Sensing Grid Patterns.

Atmel. (2011). Buttons , Sliders and Wheels - Sensor Design Guide, 72.

Axelson, J. (2007). *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*. *Lakeview Research* (2a ed.). Lakeview Research: 11-27: 9781931448062.

Azoteq. (2013). Application Note : AZD068 Trackpad Design Guide. *ProxSense Series*, 1–19.

Azoteq. (2014). Application note : AZD036 Mutual Capacitance Button Layout Guideline. *ProxSense Series*, 1–9.

Barrett, G., & Omote, R. (2010). Projected-capacitive touch technology. *Information Display*, *26*, 16–21.

Biobserve. (s/f). Behavioral Research Elevated Plus Maze with realtime data analysis and group statistics. Recuperado el 28 de mayo de 2017, a partir de http://www.biobserve.com/behavioralresearch/products/viewer/plug-ins/elevated-plus-maze/

Bohn, B. (2009). Microchip CTMU for Capacitive Touch Applications. *Technology*, 1–22.

Campden Instruments Ltd. (s/f). Elevated Plus Maze - Products and Categories. Recuperado el 28 de mayo de 2017, a partir de http://campdeninstruments.com/products/auto-elevated-plus-maze-rats

Chacón Matú, D. R. (2016). *Actualización de un sistema para el registro y análisis de la actividad motora en ratas*. Tesis de Licenciatura, Universidad Autónoma de Yucatán, Mérida, Yucatán.

Collí Alfaro, J. G. (2014). *Diseño e implementación de un sistema de monitoreo automatizado de un laberinto en "Y" para estudios de memoria en ratas*. Tesis de Licenciatura, Universidad Modelo, Mérida, Yucatán.

Davis, M. (2001). Fear-Potentiated Startle in Rats. En *Current Protocols in Neuroscience*. John Wiley & Sons, Inc. https://doi.org/10.1002/0471142301.ns0811as14

de Kloet, E. R., Oitzl, M. S., & Joëls, M. (1999). Stress and cognition: are corticosteroids good or bad guys? *Trends in Neurosciences*, *22*, 422–426. https://doi.org/10.1016/S0166-2236(99)01438-1

ECMA. (2013). *ECMA-404: The JSON Data Interchange Format*. Recuperado a partir de http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

Gonzalez, L. E., & File, S. E. (1997). A five minute experience in the elevated plus-maze alters the state of the benzodiazepine receptor in the dorsal raphe nucleus. *The Journal of Neuroscience*, *17*, 1505–1511.

Graeff, F. G. (2007). Anxiety, panic and the hypothalamic-pituitary-adrenal axis. *Revista Brasileira de Psiquiatria*, *29*, 3–6.

Harvard Apparatus. (s/f). SMART Video Tracking System. Recuperado a partir de https://www.harvardapparatus.com/webapp/wcs/stores/servlet/product_11051_10001_60201_-1_HAI_ProductDetail___

Hristov, L. (2011). Enabling Capacitive Touchscreens to Determine Touch Objects. *Atmel Corporation*, *8*, 1–4.

Intersil Corporation. (2007). SPI Protocol and Bus Configuration of Multiple DCPs, 1–5.

Lister, R. G. (1987). The use of a plus-maze to measure anxiety in the mouse. *Psychopharmacology*, *92*, 180–185. https://doi.org/10.1007/BF00177912

McEwen, B. S., & Sapolsky, R. M. (1995). Stress and cognitive function. *Current*

Opinion in Neurobiology, *5*, 205–216. https://doi.org/10.1016/0959-4388(95)80028-X

Panlab Technology for Bioresearch. (s/f). Elevated Plus Maze.indd. Recuperado el 28 de mayo de 2017, a partir de http://www.novalabcientifica.com.br/arquivos/palestra_download/Elevated Plus Maze.pdf

Patel, P. D., Seasholtz, A. F., & Patel, P. D. (2006). Computer-assisted scoring of the elevated plus maze. *BioTechniques*, *41*, 700–704. https://doi.org/10.2144/000112318

Perme, T. (2007a). AN1101, Introduction to Capactive Sensing. *Technology*, 1–10. Recuperado a partir de http://www.eetimes.com/design/industrial-control/4371081/An-introduction-to-Capacitive-Sensing-Part-I

Perme, T. (2007b). AN1102, Layout and Physical Design Guidelines for Capacitive Sensing. *Technology*, 1–10. Recuperado a partir de http://www.eetimes.com/design/industrial-control/4371081/An-introduction-to-Capacitive-Sensing-Part-I

Perme, T. (2007c). *AN1103, Software Handling for Capacitive Sensing*. *Technology*. Recuperado a partir de http://www.eetimes.com/design/industrial-control/4371081/An-introduction-to-Capacitive-Sensing-Part-I

Philips Semiconductors. (2000). The I2C-bus specification. *Philips Semiconductors*, *9397*, 954. https://doi.org/0-471-96268

Sahgal, A. (1993). *Behavioural Neuroscience: A Practical Approach* (Vol. II). IRL Press: 9780199633685. Recuperado a partir de http://books.google.com.mx/books?id=hN1qAAAAMAAJ

Salazar Loría, D. E. (2012). *Control de motores de pasos desde un ambiente de instrumentación virtual por medio de una interfaz de comunicación USB*. Tesis de Licenciatura, Universidad Autónoma de Yucatán, Mérida, Yucatán.

Schneiderman, N., Ironson, G., & Siegel, S. D. (2005). Stress and health: psychological, behavioral, and biological determinants. *Annual review of clinical psychology*, *1*, 607–28. https://doi.org/10.1146/annurev.clinpsy.1.102803.144141

Sidor, M. M., Rilett, K., & Foster, J. A. (2010). Validation of an automated system for measuring anxiety-related behaviours in the elevated plus maze. *Journal of Neuroscience Methods*, *188*, 7–13. https://doi.org/10.1016/j.jneumeth.2010.01.021

Torres, C., & Escarabajal, M. . (2002). Validation of a behavioral recording automated system in the elevated plus-maze test. *Life Sciences*, *70*, 1751–1762. https://doi.org/10.1016/S0024-3205(02)01476-5

Vera, E. de J., Heredia, F. J., Góngora, J. L., Bata, J. L., & Álvarez, F. (2013). Automatización de un laberinto en cruz basado en sensado capacitivo. En *Congreso Nacional de Ingeniería Eléctrica y Electrónica del Mayab*. 1665-0271

Walf, A. A., & Frye, C. A. (2007). The use of the elevated plus maze as an assay of anxiety-related behavior in rodents. *Nature protocols*, *2*, 322–8. https://doi.org/10.1038/nprot.2007.44

Walker, G. (2014). Part 1 : Fundamentals of Touch Technology. *SID Display Week*.

12. ANEXOS

Los anexos se encuentran en formato digital en el CD adjunto a este trabajo. Como anexos se incluye lo siguiente:

- Tabla comparativa de la imagen de los sensores y el video del experimento realizado.
- Código en lenguaje ensamblador del microcontrolador.
- Código en lenguaje C++ de la interfaz gráfica realizada con QT.
- Archivo JSON del experimento realizado