Dispositivo de Interfaz Humana

Trabajo Final de la carrera Ingeniería Electrónica

***JUAN IGNACIO PERRONE ORSI**, Legajo: 5756*

***LUCAS EZEQUIEL BORRACCI,** Legajo: 6634*

UNIVERSIDAD NACIONAL DE MAR DEL PLATA

FACULTAD DE INGENIERÍA

Fecha: 17/02/2020

Dispositivo de Interfaz Humana

Trabajo Final de la carrera Ingeniería Electrónica

***JUAN IGNACIO PERRONE ORSI**, Legajo: 5756*

***LUCAS EZEQUIEL BORRACCI,** Legajo: 6634*

UNIVERSIDAD NACIONAL DE MAR DEL PLATA

FACULTAD DE INGENIERÍA

Fecha: 17/02/2020

Índice

1.Resumen

En el siguiente informe se explica el funcionamiento de un dispositivo cuyo objetivo es actuar como interfaz entre un individuo que presente un cuadro de cuadriplejia y un sistema operativo de una PC. Su misión principal consiste en complementar las limitaciones provocadas por la lesión, de forma que le permita al usuario acceder de una manera sencilla a las funcionalidades propias de dicho sistema.

Una persona en estado de cuadriplejia solo puede realizar movimientos desde el cuello hacia arriba, es por este motivo que todo el funcionamiento del dispositivo debe estar controlado a partir de alguna acción que surja de este sector del cuerpo del usuario. Con el fin de encontrar una solución al manejo del cursor en la pantalla se propone ubicar un sensor en la nuca del usuario. A partir de los movimientos de su cabeza, el usuario puede controlar el cursor de forma cómoda y sin esfuerzo. Para solucionar el problema de las funciones propias del mouse, como por ejemplo click izquierdo, click derecho, etc., se propone la utilización de un sensor de presión. A partir de una combinación de soplidos y succiones realizados por el usuario, captados por dicho sensor, y procesados por un algoritmo, se puede tener acceso a todas las funciones propias de un mouse.

A su vez, se incluye un software con una interfaz gráfica e interactiva, a partir de la cual el usuario puede interactuar con el sistema y realizar modificaciones en su funcionamiento, personalizándolo para su conveniencia.

Con el objetivo de brindar robustez física al dispositivo**,** a la vez de proporcionar un ajuste ergonómico que permita adaptar el sistema a cualquier usuario, se diseña un soporte que se encontrará ubicado en los hombros del mismo. Al dispositivo se le dio el nombre de **InterMOUSE**.

2.Introducción

Este desarrollo surge a partir de una idea original de **Pablo Fernández**, una persona que se encuentra en la condición de cuadriplejia. Su idea consiste en controlar el cursor del mouse con movimientos de la cabeza del usuario y realizar los clicks a partir de soplidos y succiones. Un primer prototipo funcional se diseñó en base a sus propuestas y necesidades.

En el primer modelo se trabajó sobre un mouse comercial, el cual fue modificado para adaptarlo a las condiciones propias del proyecto. Por medio del uso del sensor óptico incluido en el mouse se obtuvo información acerca de la posición del cursor en la pantalla. El sensor se encontraba ubicado a la altura de la nuca del usuario, de forma que este pueda desplazarse por la pantalla mediante el movimiento de su cabeza. Los pulsadores físicos de los botones del mouse fueron reemplazados por señales provenientes del microcontrolador Atmega328P. El microcontrolador fue programado para recibir, digitalizar y analizar la señal analógica del sensor de presión MPXV7007. Este sensor, a través de una boquilla, se encargaba de detectar soplidos y succiones provocados por el usuario, y de convertirlos a valores de voltaje.

Luego del primer prototipo se decidió cambiar el enfoque del proyecto y diseñar una nueva versión. En parte el cambio fue motivado por ciertas limitaciones del anterior prototipo para ser replicado fácilmente por cualquier individuo que lo necesite. En este caso, sin realizar ninguna modificación sobre un mouse comercial, se diseñó un dispositivo independiente que puede interactuar con cualquier mouse para recibir la señal óptica de movimiento del cursor. El mouse comercial sigue ubicándose en la nuca del usuario con el fin de realizar el movimiento del puntero en la pantalla. Este segundo prototipo continúa utilizando un sensor de presión para detectar soplidos y succiones, y convertirlos en acciones del mouse. A su vez, se le añade la funcionalidad inalámbrica al dispositivo para facilitar su utilización. Otra novedad se encuentra en el software incluido en el sistema, a partir del cual el usuario puede realizar modificaciones en su funcionamiento, personalizando el dispositivo a su conveniencia. Dentro de las opciones de personalización se encuentran: sensibilidad del movimiento del puntero, modificación de los umbrales del sensor de presión para el accionado de los clicks, entre otros.

Este informe se centra en explicar y exponer el diseño y funcionamiento de este último prototipo.

Cabe destacar que se han encontrado antecedentes de mouse que intentan satisfacer la misma necesidad que **InterMOUSE.** Existen, por ejemplo, variantes que utilizan el movimiento de la pupila para el control del cursor en la pantalla, e incluso algunos utilizan la cámara web de la PC y una etiqueta en la frente del usuario para realizar los movimientos del puntero. No se han encontrado otros antecedentes de un mouse destinado a capacidades especiales que se base en un mouse comercial para el movimiento del cursor y un sensor de presión para realizar las acciones de clicks.

3.Observaciones previas y solución propuesta

3.1 Funcionamiento de un mouse

Un mouse es un dispositivo que cumple la función de interfaz entre un usuario y una PC. Para ello el usuario debe hacer uso de una de sus manos para controlar el dispositivo. Las funciones básicas de un mouse son: click izquierdo, click derecho, scroll arriba, scroll abajo, botón de scroll y movimiento del puntero en la pantalla. Este dispositivo se conecta a una PC por medio de USB y puede ser cableado o inalámbrico. Un mouse comercial típico se puede ver en la *Figura 1.*

Figura 1: Mouse comercial típico.

Otra forma de interfaz con una PC es un ***mouse pad***, que por lo general se encuentran incorporados en las computadoras portátiles o notebooks. Este se puede ver en la *Figura 2.*

Figura 2: Mouse pad.

En ninguno de estos casos existe la posibilidad de uso por parte de una persona en estado de cuadriplejia. Si bien hay variaciones de los modelos expuestos, todos necesitan realizan un movimiento con las manos. Por lo tanto, para el diseño de un dispositivo destinado a un individuo con una condición como la cuadriplejia se hace necesaria otra forma de conseguir los impulsos que generarán las acciones en el mouse.

3.2 Cuadriplejia

La cuadriplejia se trata de un estado clínico en el cual se produce una parálisis total o parcial en brazos y piernas. Esto puede ser causa de alguna lesión en las vértebras cervicales (primeras siete vértebras de la columna) o debido a una enfermedad que afecte directamente a las neuronas motoras. Las personas en este estado necesitan de una silla de ruedas para su traslado y solo están en condiciones de efectuar movimientos de la nuca hacia arriba. Dependiendo de en cuál de las vértebras cervicales se encuentre la lesión, puede haber más o menos movilidad. Cuanto más alta sea la lesión en las vértebras, más comprometida se encuentra la movilidad.

3.3 Solución propuesta

Incorporando un mouse comercial sin modificar, y con el uso de una combinación de soplidos y succiones a través de la boquilla deberían poder realizarse todas las funciones que ejecutan los botones de un mouse comercial típico. La forma de llevar a cabo la solución propuesta se profundizará a lo largo del informe con el objetivo de brindar una explicación acertada de la idea.

4. Desarrollo de Hardware

4.1 Sensado de la presión ejercida por el usuario

Se propone utilizar un sensor de presión para detectar los soplidos y succiones que genera el usuario. Luego de realizar una investigación se optó por el sensor MPXV7007 de la compañía **FREESCALE SEMICONDUCTOR**. Se trata de un transductor piezorresistivo especialmente diseñado para funcionar en conjunto con un microcontrolador que disponga de un conversor analógico-digital. El sensor puede visualizarse en la *Figura 3.*

Figura 3: Sensor de presión MPXV7007.

El patillaje del dispositivo es el siguiente:

1	2	3	4	5	6	7	8
N/C	V_s	Gnd	V_{out}	N/C	N/C	N/C	N/C

Solo tres de los ocho pines son utilizados por el dispositivo. La tensión de alimentación es **Vs=5V** a través del pin 2. El pin 3 se conecta a **GND**. Finalmente, el pin 4 se trata de **Vout**, y entrega una salida entre **0V** y **5V** proporcional a la presión medida en la boquilla. Como se mencionó anteriormente, este rango de tensión es ideal para ingresar directamente a un conversor analógico-digital sin necesidad de realizar ninguna conversión o modificación.

El sensor posee un rango de presión relativa útil entre -7 kPa y 7 kPa. Se encuentra especialmente diseñado para aplicaciones en sistemas HVAC (Calefacción, Ventilación y Aire Acondicionado), camas de hospital, sistemas respiratorios, etc. Teniendo en cuenta que la presión máxima que se podría realizar con un soplido es de aproximadamente 9,8KPa, este

sensor posee el rango adecuado para la aplicación, ya que solo se utilizarán soplidos y succiones leves que molesten lo menos posible al usuario.

4.2 Microcontrolador Atmega328p

Una vez decidido el sensor de presión a utilizar, es necesario digitalizar la información que entrega el sensor para poder procesarla. Con el objetivo de digitalizar una señal analógica se utiliza un conversor analógico-digital. En este caso se opta por el microcontrolador **Atmega328P**, que es suficiente para los fines de esta aplicación ya que cuenta con un conversor analógico-digital de 10 bits. El conversor asigna un valor digital entre 0 y 1023 para el rango de entrada analógico de **0V** a **5V**. En la *Figura 4* se puede apreciar un diagrama de pines del microcontrolador **Atmega328P**.

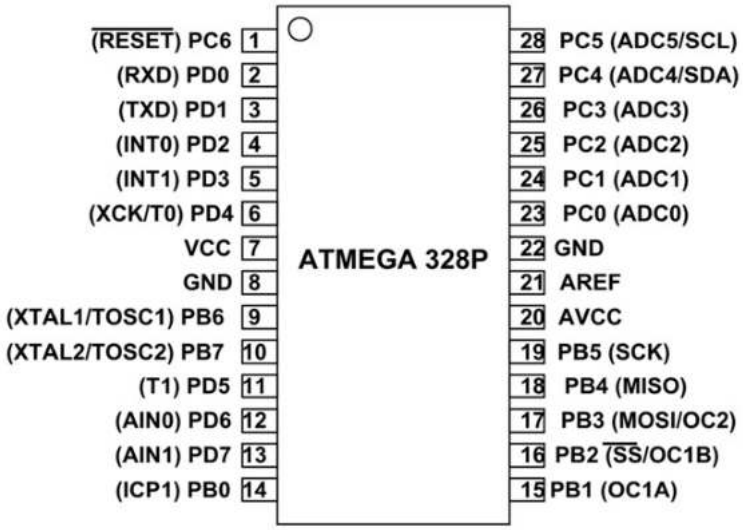

Figura 4: Microcontrolador Atmega328P.

4.3 Comunicación inalámbrica

Como se explicó anteriormente, es parte del proyecto añadir una comunicación inalámbrica al dispositivo por una cuestión de practicidad y movilidad para el usuario. De esta forma, el individuo puede ubicar la silla de ruedas libremente sin depender de la longitud de un cable. A su vez, en el caso de que el mismo se mueva de la posición de trabajo con la silla de ruedas, no es necesario desconectar y volver a realizar la conexión por parte de un tercero. Esta funcionalidad divide el dispositivo en dos partes: el transmisor (**TX**) y el receptor (**RX**).

Para realizar la comunicación inalámbrica se opta por el transceptor **NRF24L01**. Se denomina transceptor a un dispositivo que está diseñado para funcionar tanto como receptor como transmisor de una comunicación inalámbrica. Se trata de un circuito integrado económico que utiliza la banda **ISM** a **2,4GHz** con modulación **GFSK** para la transmisión inalámbrica. Existe también la capacidad de elegir diferentes velocidades de transmisión, entre los **250kbps** y los **2Mbps**. En la *Figura 5* se puede ver una imagen del encapsulado del circuito integrado **NRF24L01**.

Figura 5: Circuito Integrado NRF24L01.

En la *Figura 6* se muestra un diagrama en bloques del funcionamiento interno del integrado.

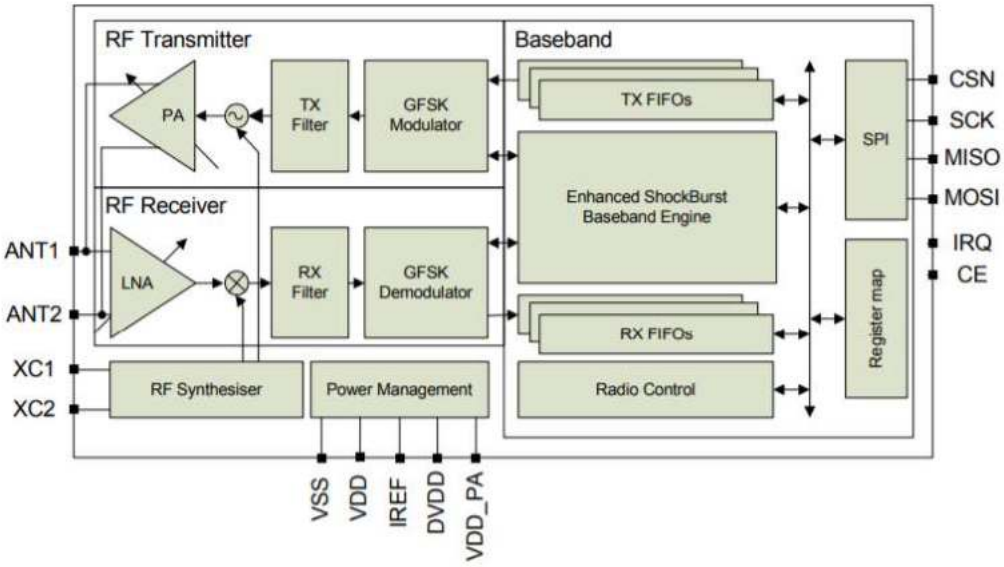

Figura 6: Diagrama en bloques del NRF24L01.

Este dispositivo hace uso de la interfaz de comunicación serie **SPI** (*Serial Peripheral Interface*) para realizar una comunicación con un microcontrolador, en este caso el **Atmega328P**.

4.4 Comunicación SPI

El protocolo SPI siempre cuenta con 1 maestro y al menos 1 esclavo. Utiliza cuatro pines para llevar a cabo la comunicación: **MOSI, MISO, SCK y CSN**. El pin **MOSI** (*Master Output Slave Input*) se utiliza para enviar información desde el MAESTRO hacia el ESCLAVO. El pin **MISO** (*Master Input Slave Output*) envía información desde el dispositivo ESCLAVO hacia el MAESTRO. El pin **SCK** (*Serial Clock*) es el reloj encargado de mantener el sincronismo en la comunicación. El pin **CSN** (*También llamado SS por Slave Selection*) es el encargado de elegir el dispositivo esclavo con el que se quiere realizar la comunicación. En este caso, el dispositivo maestro es el microcontrolador **Atmega328P** y el esclavo es el **NRF24L01**. En la *Figura 7* se puede visualizar la conexión recomendada por el fabricante en la hoja de datos del circuito integrado.

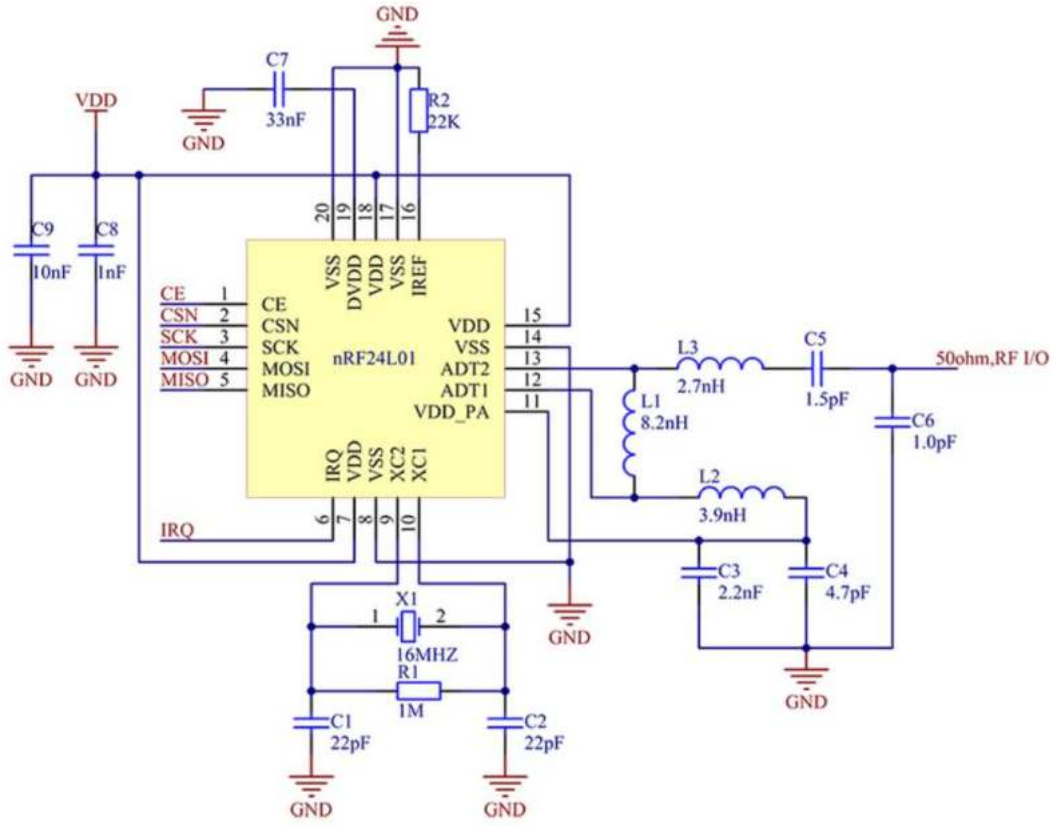

Figura 7: Esquemático de conexión de NRF24L01.

Los pines **CSN, SCK, MOSI y MISO** (*Interfaz SPI*), y **CE** (*Chip Enable*), son los que irán conectados al microcontrolador **Atmega328P**. De esta forma, el microcontrolador se comunica con el **NRF24L01** a través de la interfaz **SPI** y el módulo es el encargado de realizar la comunicación inalámbrica.

Existe un módulo comercial que integra las conexiones de la *Figura* junto con una antena diseñada en microstrip. Haciendo uso de esta antena, y según las pruebas realizadas, se consigue un alcance entre los 20 y los 30 metros, dependiendo de la velocidad de transmisión y de los obstáculos que se ubican entre el transmisor y el receptor. Este rango es suficiente para el sistema **InterMOUSE**. Este módulo resulta de gran practicidad para reducir la complejidad del montaje del hardware de esta aplicación, solo resulta necesario establecer la conexión **SPI** con el microcontrolador **Atmega328P**. En la *Figura 8* se presenta el módulo integrado de **NRF24L01+ANTENA** junto con la identificación de cada uno de sus pines.

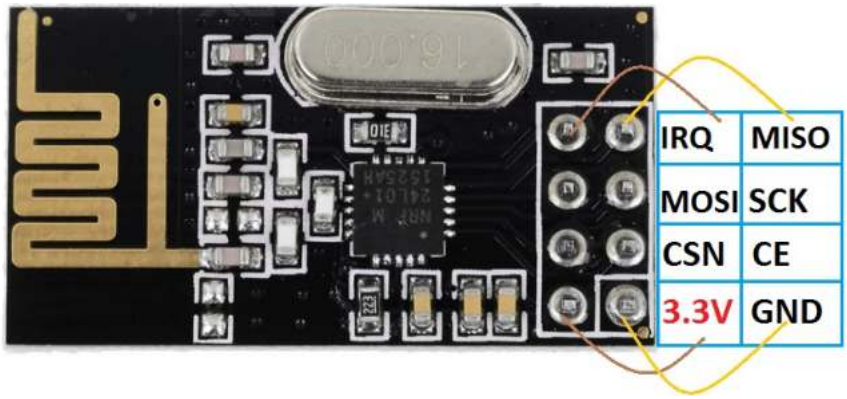

Figura 8: Módulo NRF24L01+ANTENA.

4.5 Módulo FTDI

El módulo **FTDI** funciona como transductor entre la lógica TTL de comunicación serie del microcontrolador **Atmega328p** y el protocolo USB para comunicarse con cualquier PC moderna. Es necesario conectar el pin TX del **Atmega328p** con el pin RX del **FTDI** y el pin RX del **Atmega328p** con el pin TX del **FTDI**. El **FTDI** cuenta con un conector USB Mini B, para la comunicación con la PC. Por otro lado, el **FTDI** se utiliza para alimentar el sistema receptor, ya que cuenta con salidas de **5V** y **3.3V**. En la *Figura 9* puede visualizarse el módulo en cuestión.

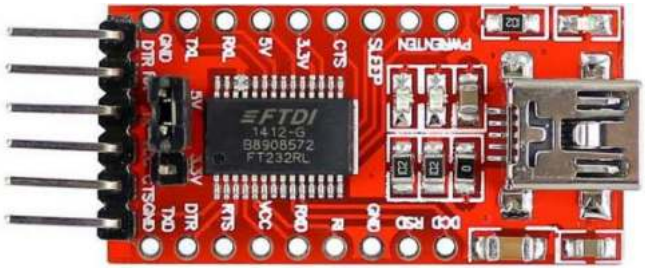

Figura 9: Módulo FTDI.

4.6 Sistema integrado

A continuación, se mostrarán en detalle los esquemáticos de los circuitos **TX** y **RX.** Estos fueron diseñados con el uso del Software **Altium Designer.**

Los elementos utilizados en el **TX** son: **1) Transductor NRF24L01**
2) Sensor de presión MPXV7007
3) Microcontrolador Atmega328P
4) Regulador 7805
5) Regulador 78R33

6) Capacitores de Bypass X 4
7) Capacitores para estabilidad del Cristal X 2
8) Cristal oscilador 16 MHz
9) Diodo de protección
10) Circuito de RESET

El esquemático del **TX** se puede visualizar en la *Figura 10.*

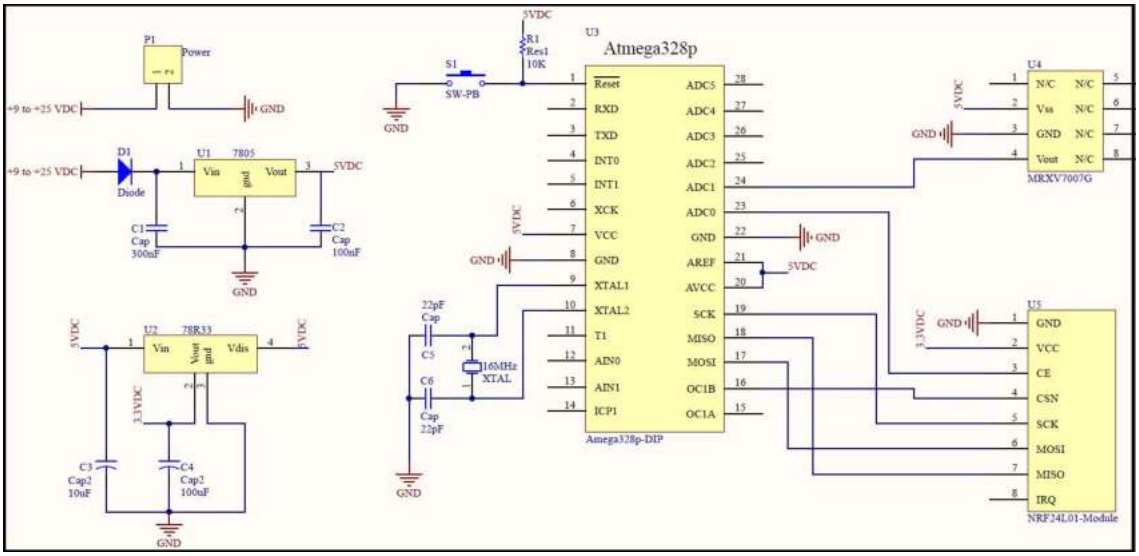

*Figura 10: Esquemático de **TX.***

Por otro lado, el circuito **RX** posee los siguientes elementos:

1) Transductor NRF24L01
2) Módulo FTDI
3) Microcontrolador Atmega328P
4) Regulador 78L33
5) Capacitores de Bypass X 2
6) Capacitores para estabilidad del Cristal X 2
7) Cristal oscilador 16 MHz

El esquemático del **RX** se puede visualizar en la *Figura 11.*

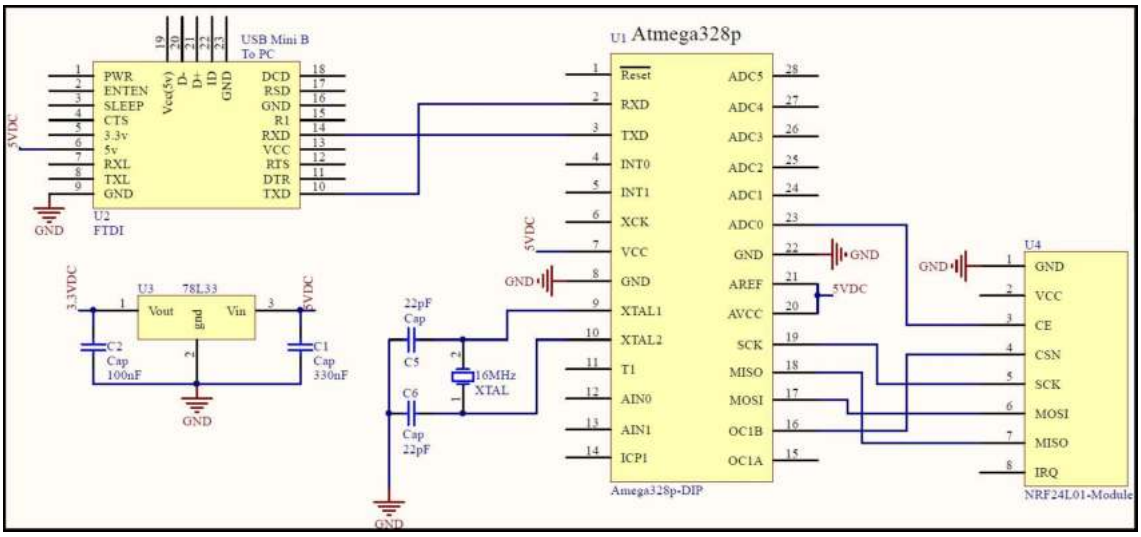

*Figura 11: Esquemático de **RX.***

4.7 Diseño de PCBs

Se procede a continuación a mostrar los diseños realizados para las placas **PCB** del sistema **InterMOUSE.** La única diferencia con los esquemáticos originales resulta que en el caso del **TX** se incluye un conector de alimentación JACK hembra de diámetro exterior **5,5mm** y diámetro interior **2,5mm**, como el que se muestra en la *Figura 12.*

Figura 12: Jack hembra 5,5mm x 2,5mm

El diseño de la **PCB** del **TX** se puede visualizar en la *Figura 13.* En color azul se muestran las pistas de la capa inferior de la placa (BOTTOM) y en color rojo las de la capa superior (TOP). También, se encuentran las dimensiones de la placa que son de aproximadamente **55mm X 54mm**.

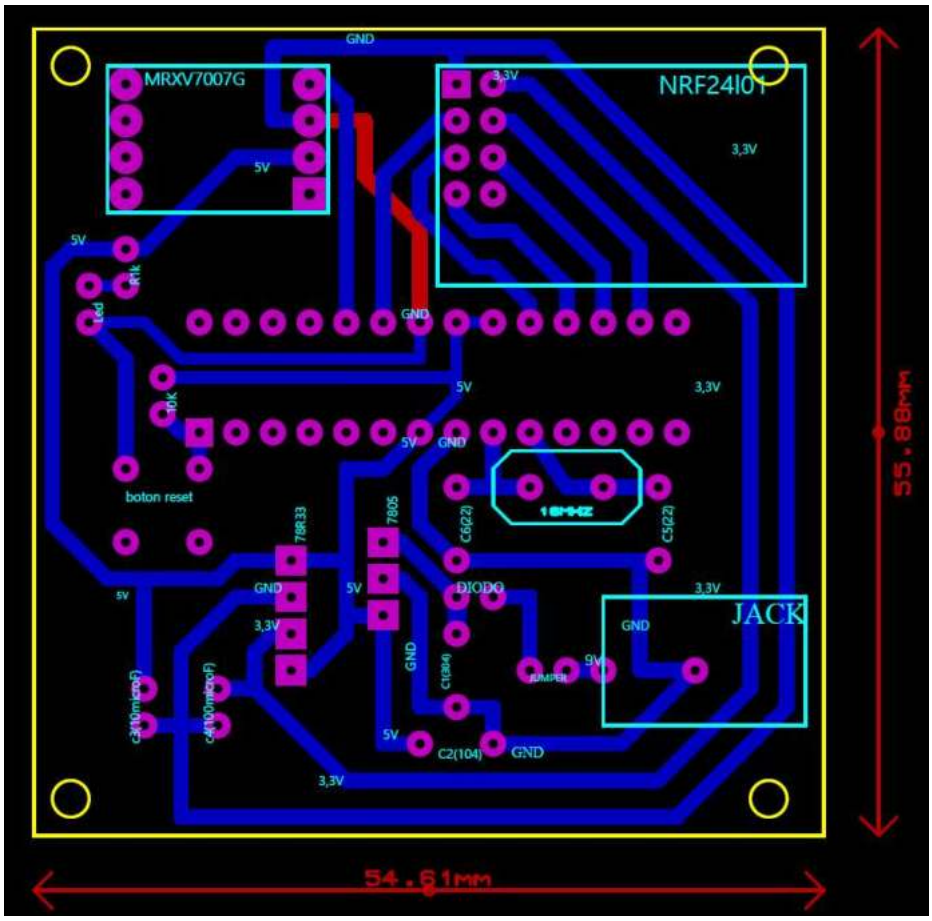

Figura 13: Diseño PCB TX

La placa PCB finalizada se muestra en la *Figura 14.*

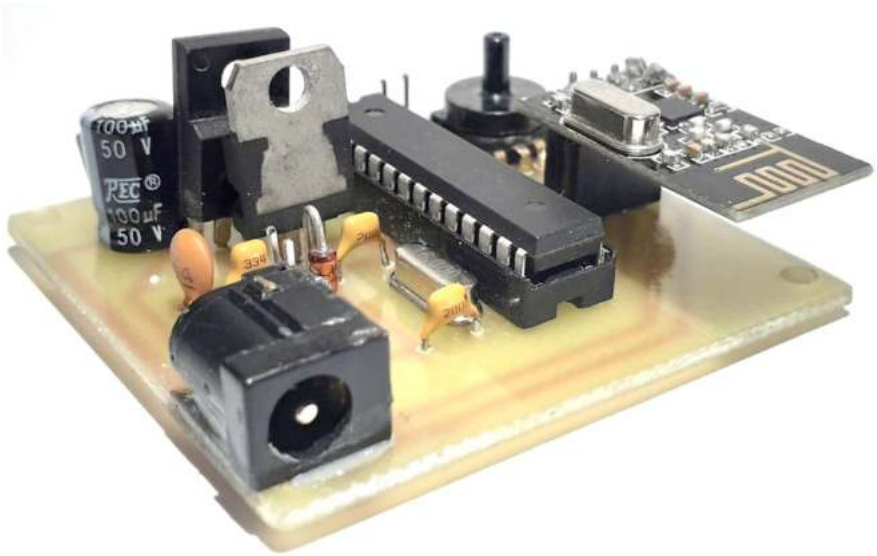

Figura 14: Placa PCB TX.

El diseño de la **PCB** de **RX** se muestra en la *Figura 15.* Las dimensiones de la placa son de aproximadamente 57mm X 48mm.

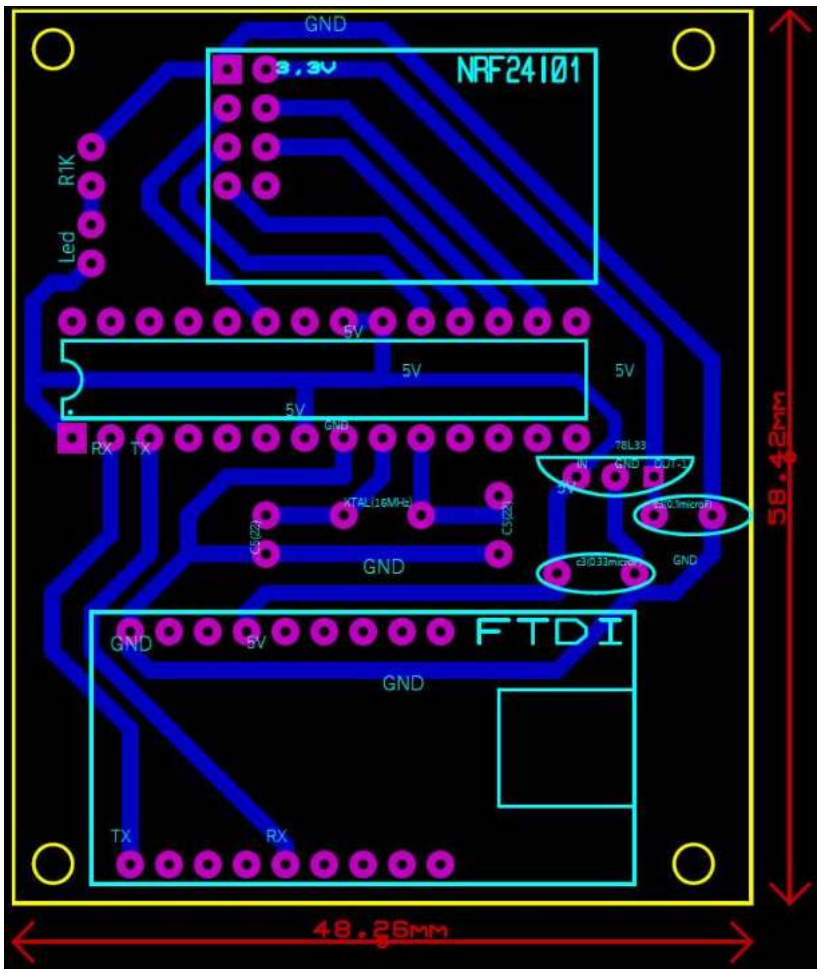

Figura 15: Diseño PCB RX.

La placa PCB finalizada se muestra en la *Figura 16.*

Figura 16: Placa PCB RX.

4.8 Autonomía y consumo

Con el objetivo de comprobar los consumos de corriente de los diseños, con un multímetro se midió la intensidad de corriente tanto de **TX** como de **RX.** El sistema se encontraba en funcionamiento nominal durante la medición. En el caso de **TX,** se midió una corriente constante de **40mA**. Por otro lado, en el **RX** se midió **42mA.**

Cabe destacar que tanto **TX** como **RX** poseen un led que consume **10mA** que se encuentra incluido en la medición. Debido al modo en que funciona el algoritmo de **InterMOUSE,** el consumo es constante debido a que se envía la información a intervalos de tiempo regular y no solo cuando hay generación de nuevos datos en el **TX.** Una posible modificación para mejorar la autonomía podría ser establecer un modo **SLEEP** en el microcontrolador **Atmega328p** para que el mismo ingrese al modo de ahorro de energía sino se encuentra en uso. A través de la generación de una interrupción por parte del sensor de presión cuando se supere determinado umbral se puede liberar al microcontrolador del modo **SLEEP,** realizar la función pertinente y luego volver al modo ahorro de energía.

En este caso se pretende alimentar el **TX** con la batería que se encuentra en las sillas de rueda eléctricas, es por eso que no se considera que este consumo sea significativo, en su defecto, también se puede alimentar al **TX** con una batería de 9V externa. Por otro lado, el **RX** se encuentra alimentado por el puerto USB de la PC.

5. Desarrollo de Software

En cuanto al software, existen dos partes bien diferenciadas. Por un lado, se encuentran el *Código Transmisor* (**cTX**) y el *Código Receptor* (**cRX**), y por el otro, el *Código para la interfaz de* ***InterMOUSE*** (**cIIM**). El **cTX** incluye la programación en lenguaje C del microcontrolador Atmega328p para la adquisición de datos y del módulo NRF24L01 para la transmisión de los mismos. El **cRX** comprende la programación en lenguaje C de otro módulo **NRF24L01** configurado para la recepción de información, la programación de un segundo microcontrolador **Atmega328p** para el procesamiento de la misma, y la conectividad con el **cIIM.** En cuanto al **cIIM,** se trata de un código diseñado en lenguaje Java, que posee una marcada característica de interfaz gráfica para que el usuario pueda configurar y personalizar el funcionamiento del sistema.

Existen dos tipos de datos que se intercambian entre las partes del sistema. El primer tipo se trata de *datos de funcionamiento (**DF**)* y el segundo tipo son *datos de configuración (**DC**)*. Los datos **DF** son aquellos que se utilizan para determinar el funcionamiento normal del sistema **InterMOUSE**, es decir todas las funciones básicas como click izquierdo, click derecho, scroll, etc. Los datos **DC** son aquellos que se utilizan para guardar la configuración realizada por el usuario al momento de la personalización.

En la *Figura 17*, puede visualizarse el esquema de intercambio de datos entre las partes.

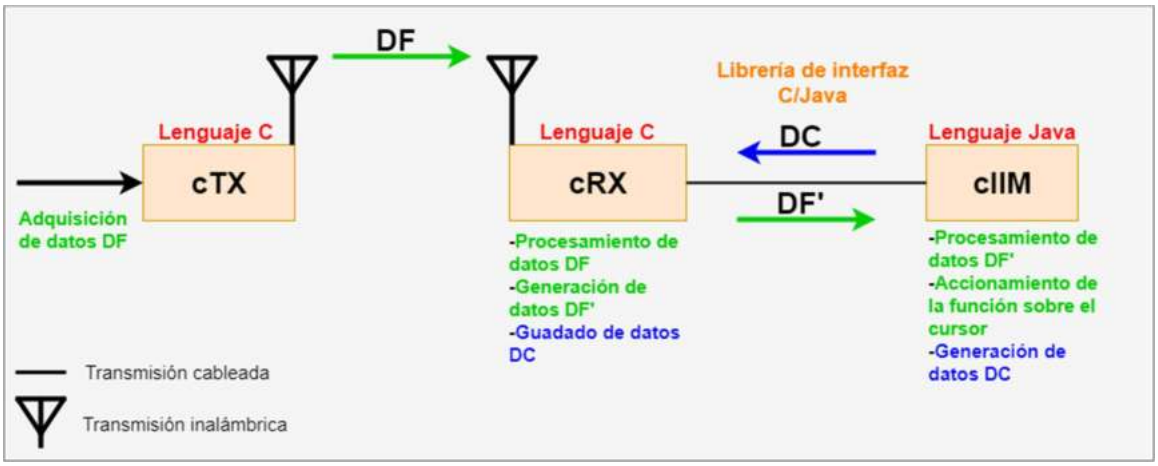

Figura 17: Esquema de intercambio de datos

El **cTX** es utilizado para adquirir los valores de presión que realiza el usuario por medio del sensor de presión. A continuación, estos datos **DF** son enviados de forma inalámbrica al **cRX.** Una vez que los datos **DF** se encuentran en **cRX** se realiza el procesamiento para convertir los datos del sensor de presión en determinadas acciones del **InterMOUSE** como el click izquierdo, click derecho, etc. Tanto el procesamiento de la información **DF** como el guardado de la información **DC** se realizan en el **cRX.** Una vez procesados los datos **DF**, estos se convierten en un nuevo tipo de datos denominados **DF'.** Seguidamente, el **cRX** se comunica a través del protocolo USB, con el **cIIM** haciendo uso de la librería de interfaz con Java denominada Panama-Hitek y envía los datos **DF'**. El **cIIM** es capaz de comprender estos datos y realizar acciones sobre al cursor en la pantalla.

Por otro lado, si el usuario lo desea puede realizar modificaciones en la interfaz de **InterMOUSE** con el objetivo de cambiar la funcionalidad o comportamiento del sistema. Una

vez terminada la configuración, el **cIIM** envía datos **DC** al **cRX** y los mismos son guardados en la memoria ROM del **cRX.** Los datos **DC** solo se intercambian entre el **cIIM** y el **cRX**.

5.1 Código Transmisor (cTX)

En la *Figura* se muestra el código del **cTX.**

```
/*-----(INCLUSIÓN DE LIBRERÍAS)-----*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
/*-----( DECLARACION CONSTANTES Y PINES )-----*/
#define CE_PIN A0
#define CSN_PIN 10
#define Ap A1
const uint64_t pipe = 0xE8E8F0F0E1LL;
/*-----( DECLARACION DE OBJETOS )-----*/
RF24 radio(CE_PIN, CSN_PIN);
/*-----( DECLARACION DE VARIABLES GLOBALES )----*/
int x, y = 0;

/*----------------------( SETUP )---------------*/
void setup() {
radio.begin();
radio.openWritingPipe(pipe);
radio.setDataRate(RF24_250KBPS);/*Se establece la velocidad
de TX a 250kbps

/*Se definen todos los pines que no se usan como salidas
  para que no funcionen como una potencial antena*/
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT);
  pinMode(A4, OUTPUT);
  pinMode(A5, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
```

```
}
/*----------------( PROGRAMA PRINCIPAL )---------*/
void loop() {
  sensadoAnalogico();
  transmisionNRF();
  delay(25);
}


/*--------------------( MÉTODOS )----------------*/
/*********************************************/
void sensadoAnalogico() {
  p = analogRead(Ap);
  x = 512;
  y = 512;
}

/*********************************************/
void transmisionNRF() {
  /*--DECLARACIÓN VARIABLES--*/
  const byte sizeTX = 6;
  byte TX[sizeTX];
  /*--CARGADO DE VECTOR--*/
  TX[0] = p / 4;
  TX[1] = x / 4;
  TX[2] = y / 4;
  /*--ENVÍO DE DATOS--*/
  radio.write(&TX, sizeTX);
}
```

A continuación, se explicará el funcionamiento del código. En primer lugar, se incluyen las librerías pertinentes para el funcionamiento del módulo **NRF24L01**. En este caso se utilizará la librería **SPI** para realizar la comunicación entre **NRF24L01** y **Atmega328p** como se indicó en la sección **4.4.** Por otro lado, se incluirá la librería RF24 para controlar la transmisión inalámbrica del módulo **NRF24L01**.

```
/*-----(INCLUSIÓN DE LIBRERÍAS)-----*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
```

En la sección del código *Declaración de constantes y pines* se definen cuáles son los pines del µC **Atmega328p** que se utilizarán para realizar la comunicación **SPI** con el módulo **NRF24L01**. Se eligen los pines, A0 como CE_PIN y 10 como CSN_PIN. Por otro lado, se define el pin analógico A1 como entrada para el sensor de presión y se lo define como Ap. Finalmente se elige el **"PIPE"** correspondiente. El **PIPE** es el canal de transmisión por software que se elige. No se debe confundir el **PIPE** con el canal de transmisión por Hardware. Los canales de Hardware son las bandas de frecuencia de aproximadamente 1MHz de ancho dentro del rango de la banda de 2,4GHz (rango entre: 2,400GHz-2,545GHz). Los **PIPES** son llamados canales lógicos en la hoja de datos del circuito integrado **NRF24L01** y constituyen los caminos por los cuales los bytes son enviados de un módulo a otro. Cada **PIPE** posee una dirección única de la forma **0xF0F0F0F0xxLL**. Donde 0x es el prefijo para indicar que el número se encuentra en hexadecimal y LL es el tipo de dato (Long Long). Según el fabricante, los primeros cuatro bytes deben tener la forma **F0F0F0F0** y luego el ultimo (xx) es arbitrario. Los **PIPES** deben tener la misma dirección en el **cTX** y en el **cRX**.

```
/*-----( DECLARACION CONSTANTES Y PINES )-----*/
#define CE_PIN A0
#define CSN_PIN 10
#define Ap A1
const uint64_t pipe = 0XF0F0F0F0E1LL;
```

Se crea un objeto de la clase RF24 con el nombre de **radio** y se le pasan los parámetros CE_PIN y CSN_PIN al constructor de la clase.

```
/*-----( DECLARACION DE OBJETOS )-----*/
RF24 radio(CE_PIN, CSN_PIN);
```

Se crean las variables **X** e **Y**, las cuales solo se utilizan para indicarle al puntero del mouse donde debe posicionarse una vez conectado el sistema **InterMOUSE.**

```
/*-----( DECLARACION DE VARIABLES GLOBALES )----*/
int x, y = 0;
```

En el **Setup** se comienza inicializando el objeto **radio** a partir de la sentencia ***radio.begin();*** y luego se debe abrir el canal lógico especificado anteriormente. Como se trata del transmisor, se debe abrir un canal de escritura a partir de la sentencia

radio.openWritingPipe(pipe);. Luego se establece la velocidad de transmisión dentro de una de las tres opciones que brinda el **NRF24L01** como se especifica en la sección **4.3.** En la última parte del **Setup**, todos los pines digitales del µC **Atmega328p** que no se usarán en el sistema se definen como salida para evitar posibles situaciones en las que los mismos puedan llegar a actuar como una antena y provocar un comportamiento erróneo en el sistema.

```
/*----------------------( SETUP )---------------*/
void setup() {
  radio.begin();
  radio.openWritingPipe(pipe);
  radio.setDataRate(RF24_250KBPS);/*Se establece la velocidad de TX a
250kbps
 /*Se definen todos los pines que no se usan como salidas
  para que no funcionen como una potencial antena*/
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT);
  pinMode(A4, OUTPUT);
  pinMode(A5, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
```

A continuación, se define el programa principal, el mismo posee dos métodos. El primer método ***sensadoAnalogico()*** se utiliza para realizar el sensado analógico del sensor de presión. El método hace uso de la función ***analogRead*** para obtener un número entre 0 y 1023 que corresponde a la presión que realiza el usuario en un determinado momento. Por otro lado, se definen los valores de las variables **X** e **Y** en 512 (la mitad del rango) para informarle al algoritmo **cIIM** que el puntero inicie en el centro de la pantalla.

El método ***transmisionNRF()*** es el encargado de enviar la información generada en esta parte del sistema. Se define un vector TX de 3 elementos y se divide a cada uno de ellos por 4, esto es así porque la librería envía datos de hasta 1 byte en cada elemento.

Se podría partir cada elemento en varios y así enviar el rango original de las variables, pero esto no es necesario, ya que la precisión que se logra con un rango de 0-255 valores es más que suficiente para los fines de la aplicación. Finalmente, se utiliza la función ***radio.write(&TX, sizeTX)*** para realizar el envío ingresando el parámetro ***&TX,*** que especifica

la dirección en memoria en que se encuentra la variable ***TX*** y ***sizeTX*** que indica el tamaño del vector.

```
/*----------------( PROGRAMA PRINCIPAL )---------*/
void loop() {
  sensadoAnalogico();
  transmisionNRF();
  delay(25);
}

/*--------------------( MÉTODOS )----------------*/
/*********************************************/
void sensadoAnalogico() {
  p = analogRead(Ap);
  x = 512;
  y = 512;
}

/*********************************************/
void transmisionNRF() {
  /*--DECLARACIÓN VARIABLES--*/
  const byte sizeTX = 3;
  byte TX[sizeTX];
  /*--CARGADO DE VECTOR--*/
  TX[0] = p / 4;
  TX[1] = x / 4;
  TX[2] = y / 4;
  /*--ENVÍO DE DATOS--*/
  radio.write(&TX, sizeTX);
}
```

5.2 Código Receptor (cRX)

El código receptor implica una mayor complejidad ya que, como se dijo anteriormente, es el encargado de guardar en memoria toda la información de configuración de **InterMOUSE.** Por otro lado, realiza la interpretación de los valores del sensor de presión y la correcta conversión a acciones del mouse, entre otras cosas.

A continuación, se muestra el código **cRX** completo, seguidamente, debido a su extensión, se explica solo las secciones más importantes del mismo.

```
/*-----(INCLUSIÓN DE LIBRERÍAS)-----*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <EEPROM.h>;   //Se incluye la librería EEPROM

/*-----( DECLARACION CONSTANTES Y PINES )-----*/
#define CE_PIN   9
#define CSN_PIN 10
```

```
const uint64_t pipe = 0xE8E8F0F0E1LL; // Define el canal de
transmisión
const byte sizeTX = 3;
const int d = 400;

/*-----( DECLARACION DE OBJETOS )-----*/
RF24 radio(CE_PIN, CSN_PIN); // Create a Radio
/*-----( DECLARACION DE VARIABLES GLOBALES )----*/
byte TX[sizeTX];
String accion = "N";
String comandoJava = "";
String stringJava = "";
String SoS = "a";
byte SopSuc;
byte Down = 29;
byte Right = 89;
byte Left = 165;
byte Up = 225;
byte p;
byte Speed;
byte Secuencia;
boolean Der = false; //variable secundaria que indica que se entro a
la zona de click derecho

/*--------------------( SETUP )----------------------*/
void setup() {
  Serial.begin(9600);
  Serial.flush();
  radio.begin();
  radio.openReadingPipe(1, pipe);
  radio.setDataRate(RF24_250KBPS);   //set datarate to 250kbps
  radio.startListening();
  cargarDatosEEPROM();
  byte memoryFlag = EEPROM.read(7);   //en caso de que no haya
ningún valor previo guardado en memoria para los UMBRALES, ingresa a
configuracionDefecto
  if (memoryFlag == B11111111) {
    configuracionDefecto();
  }
  /*SE DEFINEN COMO SALIDAS TODOS LOS PINES NO USADOS*/
  pinMode(A0,OUTPUT);
  pinMode(A1,OUTPUT);
  pinMode(A2,OUTPUT);
  pinMode(A3,OUTPUT);
  pinMode(A4,OUTPUT);
  pinMode(A5,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
}
```

```
/*---------------( PROGRAMA PRINCIPAL )---------------------------
-----------------------------------------*/
void loop() {
  recepcionNRF();
  acciones();
  transmisionSerie();
}

/*---------------- (MÉTODOS) ---------------------------------------
-----------------------------------------*/
void cargarDatosEEPROM() {
  Left = EEPROM.read(0);    //se leen los valores guardados en
memoria de los parametros del mouse y
  Right = EEPROM.read(1);  //se remapean de 255 a 1023 (ya que en
memoria se permite guardar hasta 255(1 byte))
  Up = EEPROM.read(2);
  Down =  EEPROM.read(3);
  Speed =  EEPROM.read(4);
  SopSuc = EEPROM.read(5);  // con SopSuc=1: click izquierdo
soplido, con SopSuc=0:click izquierdo succion
  if (SopSuc == 0) {
    SoS = "b";
  }
  else {
    SoS = "a";
  }
  Secuencia =  EEPROM.read(6);//Borrar
}


/**********METODO DE RECEPCIÓN DE DATOS ENVIADOS DESDE ARDUINO TX
(NRF24L01)**********************************/
void recepcionNRF() {
  if (radio.available()) {
    radio.read(&TX, sizeof(TX));
    //Serial.println(String(TX[1]) + "X" + String(TX[2]) + "Y" +
String(TX[0]) + "P" + accion);
  }
}

**METODO DE DETERMINACIÓN DE ACCIONES DEL MOUSE***********/
void acciones() {

  p = seleccionModo();
  if (p > Right && p < Left && accion != "Z") {
    accion = "N";
  }

  else if (p > Right && p < Left && accion != "B" && accion != "R")
{
    accion = "D";
  }

  else if (p >= Left && p < Up && p > Right && accion != "R") {
    accion = "I";
```

```
  }

  else if (p < Left && p <= Right && accion != "") {
    if (accion == "N") {
      retardo(d);
      recepcionNRF();
      p = seleccionModo();
    }

    if (p < Up && p < Left && p > Down && (p <= Right || p > Right)
&& accion != "B") {
      accion = "D";
    }

    else if (p < Up && p >= Left && p > Down && p > Right && accion
!= "R") {
      accion = "M";
    }

    else if (p >= Up && p > Left && p > Down && p > Right && accion
!= "") {
      accion = "R";

    }
    else if (p < Up && p < Left && p <= Down && p < Right && accion
!= "") {
      accion = "B";
    }
  }
}

/*********METODO DE SELECCIÓN DE MODO SOPSUC*************/
byte seleccionModo() {
  byte P = 0;
  P = TX[0];
  if (!SopSuc) {
    P = 255 - P;
  }
  return P;
}


/************METODO DE RECEPCIÓN DE COMANDOS JAVA***********/
void transmisionSerie() {
  lecturaString();
  comandoJava = stringJava;
  if (comandoJava == "ok") {
    enviarDatos();
  }
  else if (comandoJava == "CONDEF") {
    configuracionDefecto();
  }
  else if (comandoJava.startsWith("CONUSU")) {
    configuracionUsuario();
  }
```

```
  else if (comandoJava == "CONINI") {
    configuracionInicial();
  }
}


/*********METODO DE ENVÍO DE DATOS A JAVA**********************/
void enviarDatos() {
  Serial.println(String(TX[1]) + "X" + String(TX[2]) + "Y" +
String(TX[0]) + "P" + accion);
  Serial.flush();
}


/*---------------- (MÉTODOS DE CONFIGURACIÓN) ----------------*/
void configuracionInicial() {
  delay(200);
  cargarDatosEEPROM();
  Serial.println(String(Left) + "L" + String(Right) + "R" +
String(Up) + "U" + String(Down) + "D" + SoS);
  Serial.flush();
}

/************METODO DE CONFIGURACIÓN POR DEFECTO***************/
void configuracionDefecto() {
  //Serial.println("entró al modo configuracion por
defecto");  //BORRAR LUEGO
  byte addr = 0;
  EEPROM.write(addr, B10100101);          //addr0  // el número es
165 que representa a Left
  addr++;
  EEPROM.write(addr, B01011001);        //addr1  // el número es 89
que representa a Right
  addr++;
  EEPROM.write(addr, B11100001 );     // addr2 //el número es 225 que
representa a Up
  addr++;
  EEPROM.write(addr, B00011101);    // addr3 // el numero es el 29
que representa a Down
  addr++;
  EEPROM.write(addr, B00000100); // addr4 // velocidad 4 es el valor
por defecto
  addr++;
  EEPROM.write(addr, B00000001);      //addr5  es un valor booleano,
00000000 representa a click izquierdo como soplido
  addr++;
  addr++;
  EEPROM.write(addr, B00000000);    //addr7             //PARA EL
MEMORY FLAG// indica que se ingreso al menos 1 vez a
configuracionDefecto()
}

/******METODO DE CONFIGURACIÓN DEL USUARIO************/
void configuracionUsuario() {
  stringJava = stringJava.substring(6, stringJava.length());
```

```
  int varInt[7];
  int var;
  byte addr = 0;
  varInt[0] = (stringJava.substring(0,
stringJava.indexOf('L'))).toInt();      // java entrega valores
entre 0 y 255, solo hay que pasarlos a byte para almacenarlos
  EEPROM.write(addr, byte(varInt[0]));//addr0
  Left = varInt[0];
  addr = addr + 1;
  varInt[1] = (stringJava.substring(stringJava.indexOf('L') + 1,
stringJava.indexOf('R'))).toInt();
  EEPROM.write(addr, byte(varInt[1]));//addr1
  Right = varInt[1];
  addr = addr + 1;
  varInt[2] = (stringJava.substring(stringJava.indexOf('R') + 1,
stringJava.indexOf('U'))).toInt();
  EEPROM.write(addr, byte( varInt[2])); //addr2
  Up = varInt[2];
  addr = addr + 1;
  varInt[3] = (stringJava.substring(stringJava.indexOf('U') + 1,
stringJava.indexOf('D'))).toInt();
  EEPROM.write(addr, byte( varInt[3])); //addr3
  Down = varInt[3];

 addr = addr + 1;
  varInt[4] = (stringJava.substring(stringJava.indexOf('D') + 1,
stringJava.indexOf('S'))).toInt();  //velocidad del 0 al 6
  EEPROM.write(addr, byte(varInt[4]));//addr4
  Speed = varInt[4];
  addr = addr +
1;                                  //163L86R226U22D4SacF
  var = stringJava.indexOf('a');
  if (var != -1) {
    varInt[5] = B00000001;  //valor por defecto
    SopSuc = 1;      // soplido = click izquierdo--succion = click
derecho
  }
  if (var == -1) {
    varInt[5] = B00000000;
    SopSuc = 0;     //soplido = click derecho -- succion = click
izquierdo
  }
  EEPROM.write(addr, byte(varInt[5]));//addr5
  addr = addr + 1;
  var = stringJava.indexOf('c');//borrar determina si el tercer
botón se activa con soplido/succión (la c ES 19), o si manda -1
(quiere decir que encontró d) es succión/soplido
  if (var != -1) {//borrar
    varInt[6] = B00000000;  //borrar //valor por defecto
    Secuencia = 1; //borrar //3er boton sop/suc
  }//borrar
  if (var == -1) {//borrar
    varInt[6] = B00000001;//borrar
    Secuencia = 0; //borrar //3er boton suc/sop
  }//borrar
```

```
  EEPROM.write(9, varInt[6]); //addr8
  addr = 0;
  EEPROM.write(7, B00000000);   //addr7            //PARA EL MEMORY
FLAG// indica que se ingresó alguna vez al modo configuracionUsuario
}

/**METODO DE LECTURA DEL STRING DE DATOS ENVIADO POR JAVA******/
void lecturaString() {
  char caracter = "";
  stringJava = "";
  do {
    while (Serial.available() > 0) {
      caracter = Serial.read();
      if (caracter != '\n') {
        stringJava = stringJava + caracter ; //string de datos
mandados desde java con la configuracion de los parametros de
intermouse (255L255R255U255D7Sac\n)
      }
    }
  }
  while (caracter != '\n');
}
void retardo(int D) {
  long a = millis();
  while (millis() - a < D) {
    recepcionNRF();
  }
}
//****************FIN DEL CÓDIGO cRX*****************
```

Al igual que en el **cTX,** se incluyen las mismas librerías, con el añadido de la librería EEPROM, necesaria para guardar información en la memoria interna del microcontrolador **Atmega328p**.

```
/*-----(INCLUSIÓN DE LIBRERÍAS)-----*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <EEPROM.h>;   //Se incluye la librería EEPROM
```

Luego se define el PIPE que coincide con el que se definió para el **cTX**:

```
const uint64_t pipe = 0XF0F0F0F0E1LL; // Define el canal de
transmision
```

Una de las partes más importantes de este código es la forma en que se determinan las acciones del mouse a partir de los valores de presión recibidos, esto se explicará a continuación.

El método que realiza el procesamiento de las acciones de **InterMOUSE** se denomina ***void acciones().*** Como se dijo anteriormente, el vector TX[] posee los elementos P, X e Y. X e Y son fijos y se utilizan para inicializar la posición del puntero en el medio de la pantalla, mientras que P es un valor entre **0 y 255** que indica el valor de presión generado por el usuario. Cuando el usuario no realiza ningún soplido ni succión, el valor enviado se encuentra en la mitad del rango útil, es decir con un valor de aproximadamente **128**. Como este valor fluctúa de acuerdo a la presión atmosférica, y con el objetivo de determinar de forma correcta la acción que se está intentando ejecutar se hace necesario establecer **UMBRALES**. Se establecen **4 UMBRALES,** cada uno tiene un nombre específico y se asocia a un determinado valor de presión. Esto puede verse en la *Figura 18.*

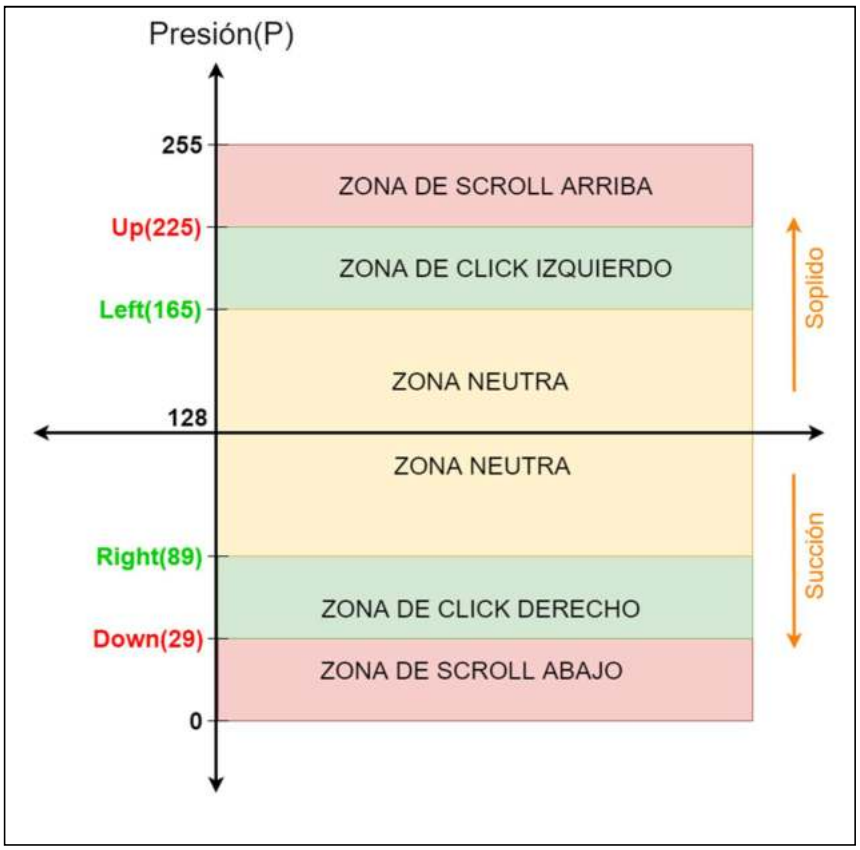

Figura 18: Mapa de umbrales y su relación con los valores de presión.

Mientras el valor de **P** se encuentre en el rango: **Right(89)<P<Left(165), InterMOUSE** se encuentra en la ZONA NEUTRA, aquí la acción se denomina ACCIÓN NULA. Si el usuario

realiza un SOPLIDO LEVE, y ubica la variable P en el rango: **Left(165)<P<Up(225),** entonces se ingresa a la ZONA DE CLICK IZQUIERDO. Esta acción es análoga a mantener el botón izquierdo del mouse apretado sin soltarlo, es por eso que la acción se denomina MANTENIMIENTO. Mientras la variable **P** se encuentre en la ZONA DE CLICK IZQUIERDO la acción continuará siendo de MANTENIMIENTO. Una vez que el usuario cese la presión, la variable **P** comenzará a decrementar su valor, hasta que en un momento caerá por debajo del umbral **Left(165),** recién en este momento se ejecuta la acción de CLICK IZQUIERDO, que es análoga a soltar el botón izquierdo del mouse. En otras palabras, estar en la ZONA DE CLICK IZQUIERDO no significa haber realizado la acción, sino que la acción en sí se ejecutará una vez que la variable P se encuentre por debajo del umbral **Left(165).**

En el caso de querer realizar una acción de DOBLE CLICK, el usuario deberá realizar la acción CLICK IZQUIERDO dos veces, es decir que deberá realizar dos SOPLIDOS LEVES, cesando la presión entre ambos soplidos para volver a la ZONA NEUTRA.

Para realizar una acción de CLICK DERECHO es necesario que el usuario realice una SUCCIÓN LEVE y que la variable **P** ingrese a la ZONA DE CLICK DERECHO superando el umbral **Right(89)**, y finalmente que cese la succión para entrar nuevamente a la ZONA NEUTRA.

El accionado de los SCROLLS es levemente diferente al funcionamiento de los CLICKS. Primero es necesario que el usuario ingrese a la ZONA DE CLICK DERECHO, y rápidamente ejecutar un SOPLIDO FUERTE o una SUCCION FUERTE. Un SOPLIDO FUERTE se da cuando la variable **P** supera el umbral **Up(225),** en este caso se realiza una acción de SCROLL ARRIBA. Si en vez de realizar un SOPLIDO FUERTE se realiza una SUCCIÓN FUERTE, entonces se ejecuta la acción de SCROLL ABAJO. Cabe aclarar que los términos SOPLIDO/SUCCIÓN FUERTE son solo representativos y que en realidad se buscó que los soplidos y las succiones que deben ser ejecutados para superar los umbrales y para manejar de forma eficiente el sistema **InterMOUSE** fuesen lo más leves posibles para no fatigar al usuario.

El método anterior se ideó debido a que existía un problema de diferenciación entre acciones de CLICK y SCROLL. Para llegar a la ZONA DE SCROLL, es necesario atravesar la ZONA DE CLICK y en ocasiones el algoritmo confundía las acciones o realizaba un CLICK luego de haber realizado el SCROLL. Con este método, una vez que se realiza una SUCCIÓN SUAVE y se ingresa a la ZONA DE CLICK DERECHO se ejecuta un DELAY de aproximadamente **400ms**. Una vez pasado el DELAY se sensa nuevamente el valor de la variable **P**: si **P** se encuentra en la

ZONA NEUTRA, se ejecuta una acción de CLICK DERECHO; si **P** se encuentra en la ZONA DE SCROLL ARRIBA, se ejecuta una acción de SCROLL ARRIBA y finalmente si **P** se encuentra en la ZONA DE SCROLL ABAJO, se ejecuta una acción de SCROLL ABAJO.

Finalmente, la acción de CLICK MEDIO, se ejecuta de forma similar a los SCROLL, con la diferencia de que luego de realizar una SUCCIÓN SUAVE e ingresar a la ZONA DE CLICK DERECHO, inmediatamente se debe realizar un SOPLIDO LEVE ingresando a la ZONA DE CLICK IZQUIERDO.

Una vez procesada la información y determinada la acción que se está ejecutando es necesario que el **cRX** le envíe esta información al **cIIM.** Para ello se utiliza el método: *void enviarDatos().*

```
/*********METODO DE ENVÍO DE DATOS A JAVA**********************/
void enviarDatos() {
  Serial.println(String(TX[1]) + "X" + String(TX[2]) + "Y" +
String(TX[0]) + "P" + accion);
  Serial.flush();

}
```

Se envía una variable del tipo String de la forma **xxXxxYxxPacción**. Donde en xxXxxY se envían los datos de la posición del cursor, como se dijo anteriormente. En xxP se envía el valor en el rango **0-255** de la variable **P** y finalmente se envía la acción que se ha determinado. Cada acción está asociada a un caracter con el fin de simplificar el traspaso de información. En la *Figura 19* se visualiza lo anterior.

ACCIÓN	CARACTER
CLICK IZQUIERDO	"I"
CLICK DERECHO	"D"
SCROLL ARRIBA	"R"
SCROLL ABAJO	"B"
CLICK MEDIO	"M"
ACCIÓN NULA	"N"

Figura 19: Tabla de acciones disponibles junto con el caracter que las representa

Por ejemplo, el envío de una sucesión de Strings para realizar un click izquierdo sería el siguiente:

La siguiente secuencia sucede en un lapso de tiempo de aproximadamente 500ms.

128X128Y128PN -------> ZONA NEUTRA/ACCIÓN NULA (sin soplido/succión)

128X128Y130PN -------> ZONA NEUTRA/ACCIÓN NULA (el usuario comienza a realizar un soplido)

128X128Y135PN -------> ZONA NEUTRA/ACCIÓN NULA (El usuario continúa el soplido, P continúa aumentando)

128X128Y147PN -------> ZONA NEUTRA/ACCIÓN NULA (P continúa aumentando)

128X128Y150PN -------> ZONA NEUTRA/ACCIÓN NULA (P continúa aumentando)

128X128Y157PN -------> ZONA NEUTRA/ACCIÓN NULA (P continúa aumentando)

128X128Y163PN -------> ZONA NEUTRA/ACCIÓN NULA (P continúa aumentando)

128X128Y168PI -------> ZONA CLICK IZQUIERDO/ACCIÓN MANTENIMIENTO (P supera el umbral Left(165))

128X128Y175PI -------> ZONA CLICK IZQUIERDO/ACCIÓN MANTENIMIENTO (P continúa aumentando)

128X128Y170PI -------> ZONA CLICK IZQUIERDO/ACCIÓN MANTENIMIENTO (El usuario cesa el soplido, P comienza a decrementar)

128X128Y167PI -------> ZONA CLICK IZQUIERDO/ACCIÓN MANTENIMIENTO (P continúa decrementando)

128X128Y160PN -------> ZONA NEUTRA/ACCIÓN CLICK IZQUIERDO (P supera nuevamente el umbral Left(165) y recién en este momento se realiza un click izquierdo)

128X128Y145PN -------> ZONA NEUTRA/ACCIÓN NULA (sin soplido/succión)

128X128Y137PN -------> ZONA NEUTRA/ACCIÓN NULA (sin soplido/succión)

128X128Y128PN -------> ZONA NEUTRA/ACCIÓN NULA (sin soplido/succión)

Otra sección importante del **cRX** es la del guardado de la información de configuración. Existen dos formas de configuración, la CONFIGURACIÓN POR DEFECTO y la CONFIGURACIÓN DE USUARIO. Cada una de ellas es accesible a través de la interfaz gráfica que existe en el **cIIM,** el cual se explicará más adelante. Por ahora solo es necesario decir que el **cRX** recibe cierta información que debe guardar. El método que recibe la información de la configuración se denomina ***void transmisionSerie()***

Cuando se desea realizar una CONFIGURACIÓN DE USUARIO, el **cIIM** envía el comando **"CONUSU".**

```
void transmisionSerie() {
  lecturaString();
  comandoJava = stringJava;
  if (comandoJava == "ok") {
    enviarDatos();
  }
  else if (comandoJava == "CONDEF") {
    configuracionDefecto();
  }
  else if (comandoJava.startsWith("CONUSU")) {
    configuracionUsuario();
  }
  else if (comandoJava == "CONINI") {
    configuracionInicial();
  }
}
```

Luego el **cIIM** envía la información al **cRX** en el formato: xxLxxRxxUxxDa/b. La sección xxLxxRxxUxxD se trata de los valores de los cuatros umbrales: LEFT, RIGHT, UP, DOWN. Estos valores se encuentran en el rango de **0-255**. Por defecto, el sistema **InterMOUSE** posee los valores **LEFT= 165, RIGHT=89, UP=225, DOWN=29**, que son aquellos que los diseñadores han encontrado más acertados para un buen funcionamiento. Sin embargo, cada usuario tiene la posibilidad de personalizar esos umbrales para modificar la cantidad de presión que se necesita realizar para efectuar una acción. El último caracter **a/b** se utiliza para determinar si el usuario desea invertir la funcionalidad de SOPLIDO/SUCCIÓN. Por defecto cuando se realiza un SOPLIDO se genera un CLICK IZQUIERDO y cuando se realiza una SUCCIÓN se genera un CLICK DERECHO, pero el usuario tiene la posibilidad de invertir esta secuencia. Si se envía el caracter **a** se utiliza la secuencia por defecto y si se envía **b** se invierte esta secuencia.

Una vez enviada esta información el algoritmo la guarda en la memoria ROM del microcontrolador **Atmega328p** a partir del método ***void configuracionUsuario().***

Existe también la posibilidad de que el usuario vuelva a los valores por defecto. En este caso el **cIIM** envía el comando **"CONDEF"** y el algoritmo vuelve automáticamente a la configuración por defecto. Más adelante, en la sección del algoritmo **cIIM,** se explicará cómo el usuario tiene acceso a estas configuraciones.

5.3 Código para la interfaz de InterMOUSE (cIIM)

Una vez que las señales provenientes del sensor de presión fueron sensadas por el transmisor e interpretadas por el receptor, resta definir un método para que la PC responda a dichas acciones. Para ello se creó un programa en lenguaje Java mediante el entorno de desarrollo **NetBeans**, que no sólo permite interpretar los datos enviados por el **cRX** sino que además proporciona herramientas de calibración para darle un ajuste fino al usuario en términos de niveles de sensibilidad del sensor de presión.

El programa en sí se encuentra constituido por cinco partes, cada una encargada de una cierta cantidad de tareas.

La primera parte del programa que puede ser apreciada en pantalla es una ventana llamada "Inicio". Su propósito es hacerle saber al usuario que el programa se está iniciando, y además invoca a la siguiente parte del programa. Es una ventana simple, que muestra el logo del proyecto durante un tiempo determinado.

La segunda parte es la principal, y se muestra como un ícono en el Área de Notificación de Windows llamado "Residente". Se encarga de nuclear las funciones más importantes del programa, tales como iniciar la conexión entre la PC y el Receptor, abrir la ventana de configuración, interpretar los mensajes recibidos por el puerto serie USB y provocar las acciones de los botones del mouse, entre otras.

La tercera parte se encarga específicamente de la conexión entre la PC y el Receptor, y se llama "Conexion". Ejecuta tareas de interfaz, envío y recepción de datos con el puerto serie USB. Es invocada principalmente por la clase Residente, y también se dedica a darles formato a los datos para que puedan ser enviados correctamente.

La cuarta parte es una ventana llamada "Frame", que permite visualizar y reconfigurar los parámetros de calibración del sensor de presión, así como cambiar el comportamiento del Receptor a la hora de interpretar los datos. Es posible acceder a esta ventana a través de un menú que muestra el ícono de Residente al hacer un click derecho y seleccionar "Abrir ventana de configuración". En la parte superior de la ventana se muestran los distintos niveles que habilitan los clicks izquierdo y derecho, y los movimientos de rueda del mouse o scroll. A la derecha se puede apreciar una barra de progreso que muestra el valor actual del sensor de presión. Por debajo del área de configuración de los niveles, se hallan dos botones. Estos permiten seleccionar el modo de operación del InterMOUSE, es decir, si un soplido realizado corresponde a un click izquierdo o derecho. Finalmente, en la parte inferior de la ventana se

pueden apreciar cuatro botones. El primero de ellos, que dice "Conectar", se utiliza para elegir el puerto correcto para la conexión serie, en caso de ser necesario (el programa detecta automáticamente todos los puertos COM al iniciar, y se conecta al último de ellos para evitar el puerto COM1 que suele ser el puerto serie original en algunas PCs). El segundo botón dice "Resetear" y su función es llevar al Receptor a su configuración inicial, revirtiendo los parámetros de niveles y comportamiento que hayan sido modificados previamente. El tercer botón dice "Grabar", y se encarga de programar el Receptor con los parámetros que se hallen en la ventana en ese momento. Estos parámetros son grabados en la memoria EEPROM del Atmega328P, y son cargados cada vez que el Receptor se inicia. Esto quiere decir que los datos personalizados seguirán siendo válidos sin importar cuántas veces se haya desconectado el Receptor de su fuente de alimentación. Finalmente, el cuarto botón se encarga de cerrar toda la aplicación. Cabe destacar que, si la ventana se cierra con la cruz superior derecha, el Residente seguirá funcionando en segundo plano.

La última ventana se llama "Puertos" y permite conectarse a otro puerto COM, en el caso de encontrarse vinculado a un puerto equivocado. En esta ventana se puede apreciar una lista con todos los puertos detectados, al igual que un botón que permite actualizar la lista y mostrar puertos que pudieran no estar presentes anteriormente.

6. Soporte InterMOUSE

Con el objetivo de brindar robustez física al Hardware del sistema **InterMOUSE,** a la vez de proporcionar un uso simple y eficiente del mismo, se realizó el **Soporte InterMOUSE (SIM).** El **SIM** fue diseñado con el uso del software **3D Studio**. El **SIM** puede visualizarse en la *Figura 20* y la *Figura 21.*

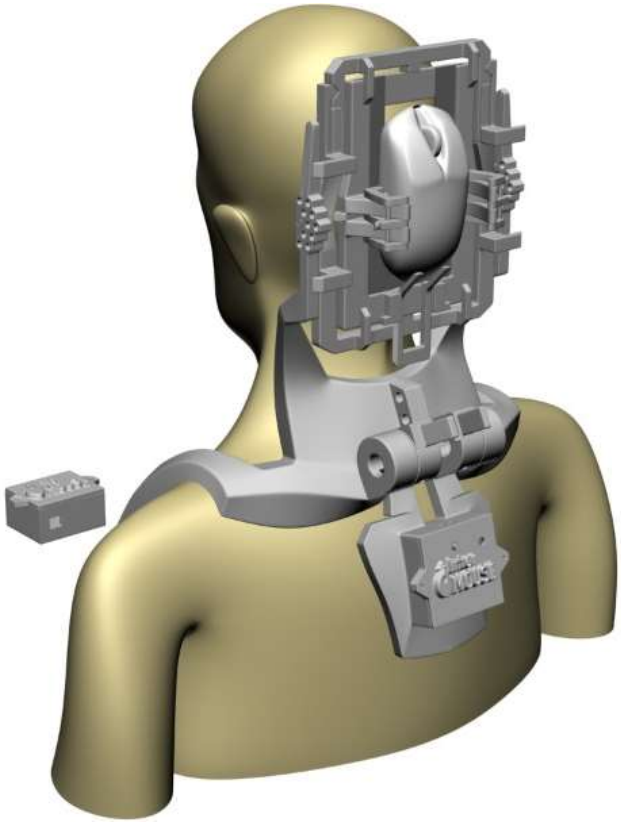

*Figura 20: Soporte **SIM** ubicado en los hombros del usuario, visto de espalda.*

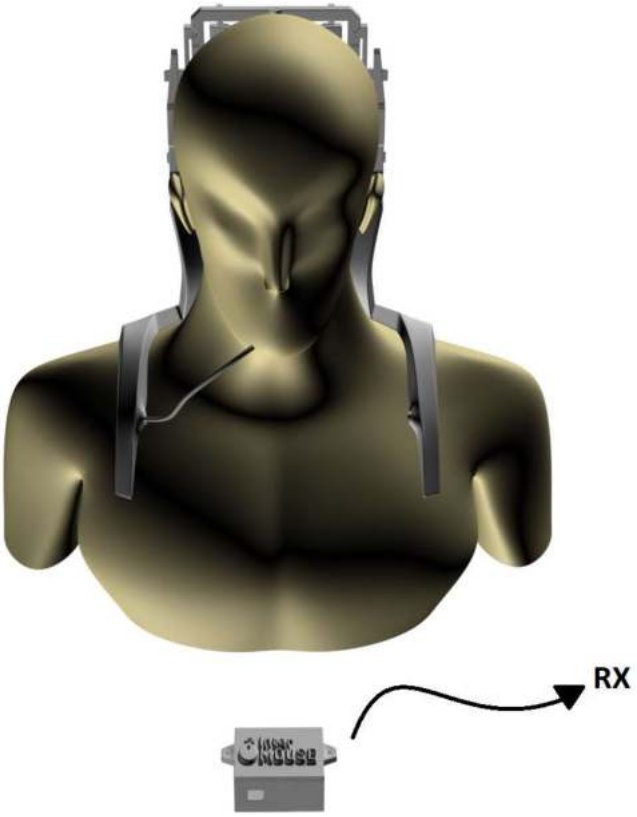

*Figura 21: Soporte **SIM** ubicado en los hombros del usuario, visto de frente.*

El **SIM** posee las siguientes funciones:

1- Ajustar de forma ergonómica el sistema InterMOUSE a cualquier usuario del mismo.
2- Proporcionar un sostén físico para el TX.
3- Proveer un agarre para el mouse comercial que se utiliza para adquirir la señal de movimiento.
4- Ajustar el mouse comercial a una posición cómoda para el usuario
5- Sostener la manguera médica que conecta las acciones del usuario con el sensor de presión MPXV7007.

A continuación, se explican en detalle cada una de estas funciones junto con la sección del **SIM** que la hace posible. También se proporcionan referencias visuales de las piezas con el fin de que la explicación quede lo más clara posible.

1- **Ajustar de forma ergonómica el sistema InterMOUSE a cualquier usuario del mismo:** El **SIM** posee un sistema que permite el ajuste ergonómico a cualquier usuario. El sistema cuenta con engranajes que se traban firmemente cuando se encuentran ajustados y que permiten un movimiento rotatorio cuando se encuentran destrabados. Es necesario realizar una presión sobre las pestañas de ajuste para destrabar los engranajes. Esto permite abrir y cerrar los brazos del **SIM** con el fin de realizar una presión sobre los hombros y el torso del usuario que ajuste firmemente el soporte. Este método de ajuste puede visualizarse en la *Figura 22.* Las flechas azules indican donde debe realizarse la presión para destrabar los engranajes y poder realizar el movimiento rotatorio mostrado por las flechas rojas. Una vez cesada la presión sobre las pestañas de ajuste, el **SIM** permanecerá en esta posición.

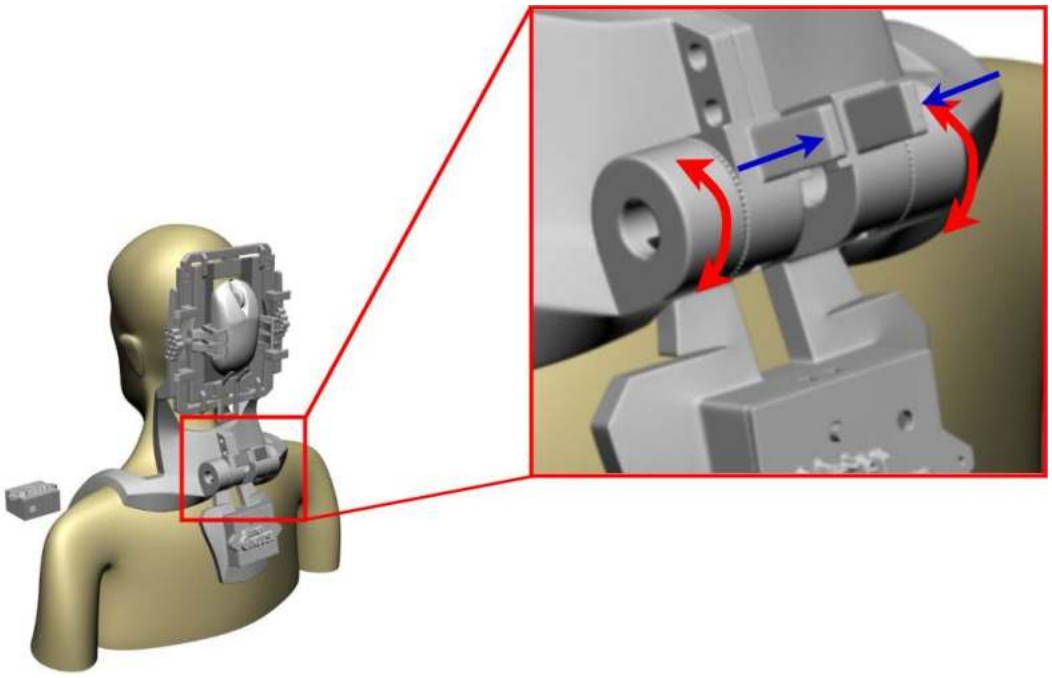

*Figura 22: Detalle del mecanismo de ajuste del **SIM.***

2- **Proporcionar un sostén físico para el TX:** El **TX** se encuentra encerrado y atornillado en un contenedor en la parte trasera del **SIM**. El contenedor posee dos aberturas. Por una abertura se desprende la manguera médica que pasa por el interior del **SIM** hasta la parte delantera del mismo para quedar cerca de la boca del usuario. Por otra abertura se conecta la alimentación del **TX** a través del Jack. El **SIM** cuenta además con un interruptor para encender el sistema **InterMOUSE** y con un LED de señalización de encendido. Esta sección del **SIM** puede visualizarse en la *Figura 23.*

Cada uno de los números indicados en el dibujo hacen referencia a:

1) **Abertura para ubicar el pulsador de encendido/apagado.**
2) **Abertura de LED de encendido/apagado.**
3) **Abertura para acceder al pulsador de Reset.**
4) **Abertura para la salida de la manguera médica.**

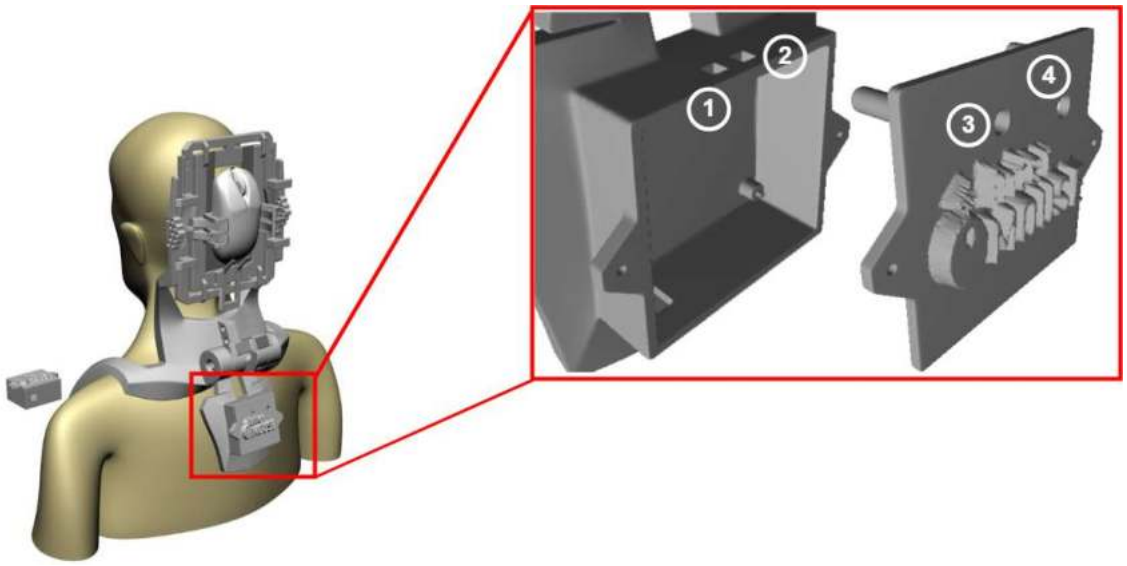

*Figura 23: Detalle del contenedor donde se aloja el **TX.***

3- **Proveer un agarre para el mouse comercial que se utiliza para adquirir la señal de movimiento:** Debido a la cantidad de tipos de mouse y a las diversas formas que poseen se diseñó un sostén lo más genérico posible para satisfacer esta función con cualquier mouse que se conecte. Al ser un agarre muy versátil, se complejiza el diseño del sostén, es por eso que se muestra un despiece de esta sección que puede ser visualizada en la *Figura 24.*

Cada uno de los números indicados en el dibujo hacen referencia a:

1) **Esta sección actúa como unión entre el sostén del mouse comercial y el resto del soporte SIM.**
2) **Base de apoyo para el mouse comercial con una abertura para la lectura del sensor.**
3) **y 4) Agarres laterales para el mouse comercial ajustados por resortes.**
5) **Tope inferior para el mouse comercial.**
6) **Mouse comercial.**

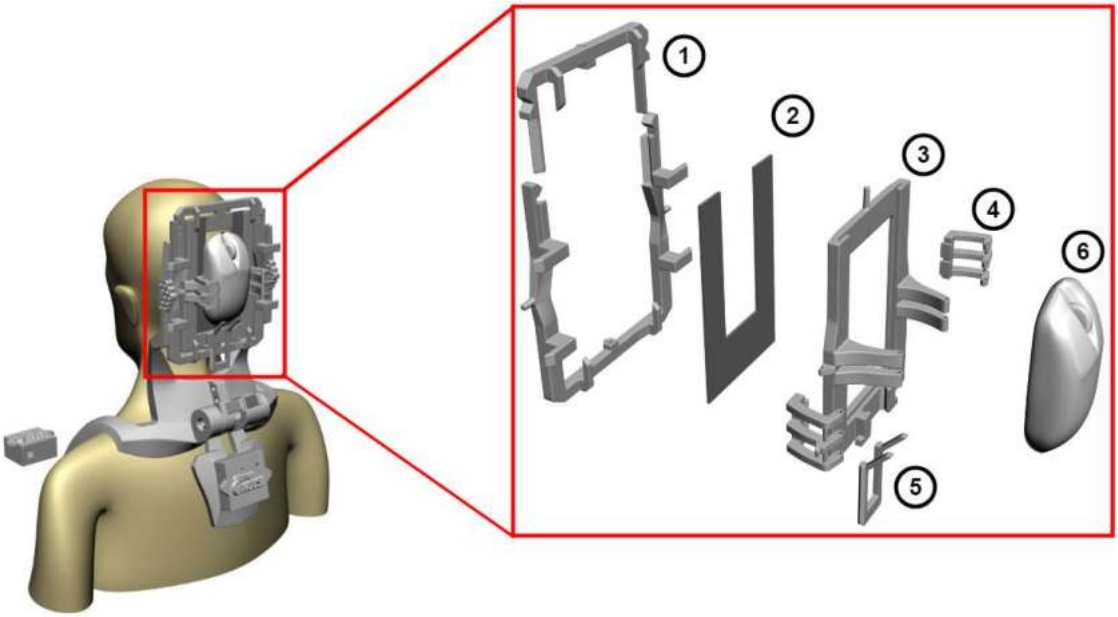

Figura 24: Detalle del sostén para el mouse comercial.

La sección 1) permite un acotado movimiento de pivote a través de un eje horizontal, y la sección 3) permite un acotado movimiento de pivote a través de un eje vertical. Esto se visualiza en la *Figura 25.* Este pivote se añade con el fin de seguir los movimientos de la cabeza del usuario de forma más fluida sin generar una incomodidad. Así, se puede decir que el sostén se adapta a los movimientos del usuario. La sección 1) y 3) se encastran gracias a la sección móvil 1.1) que también se muestra en la figura.

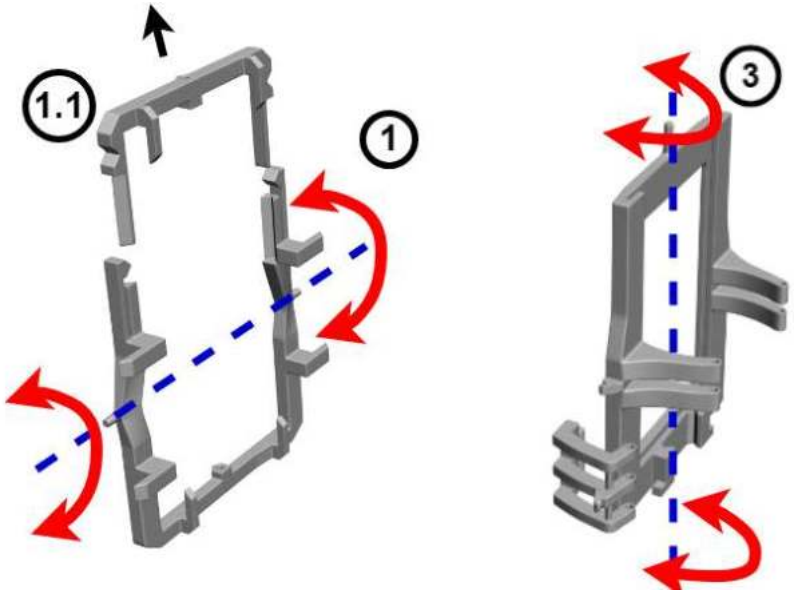

Figura 25: Detalle del movimiento de la sección 1) y la sección 3).

4- **Ajustar el mouse comercial a una posición cómoda para el usuario:** Aparte del agarre del mouse comercial, es también necesario que el mismo pueda ajustarse en la posición adecuada. Debido a que el mouse comercial debe encontrarse cerca de la nuca del usuario, con el fin de que pueda realizar los movimientos del puntero, se hace necesario que el sostén regule y mantenga la posición del mouse comercial. En la *Figura 26* se puede visualizar el sistema de encastre para posicionar el mouse de forma conveniente.

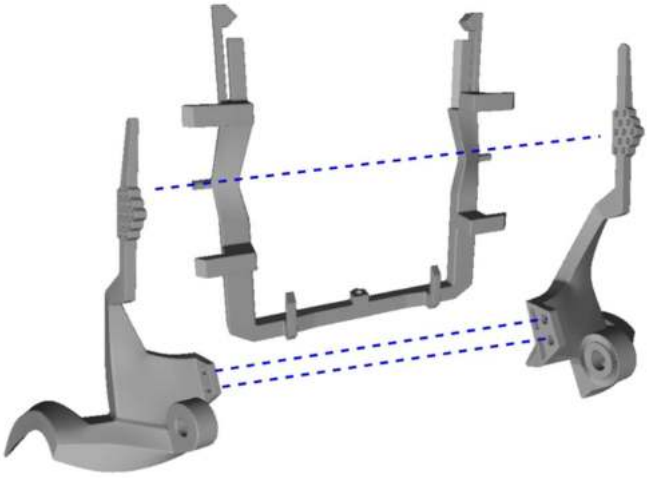

Figura 26: Sistema de encastre para posicionamiento del Mouse comercial.

5- **Sostener la manguera médica que conecta las acciones del usuario con el sensor de presión MPXV7007:** Por su parte, la manguera médica también necesita un sostén para quedar ubicada cerca de la boca del usuario. De esa forma, el mismo puede generar las acciones de soplido y succión de forma adecuada. La manguera se encuentra recubierta por un resorte y a su vez un alambre se encuentra en su interior con el fin de brindar una característica de maleabilidad y firmeza a la boquilla. Lo anterior puede verse en la *Figura 27.*

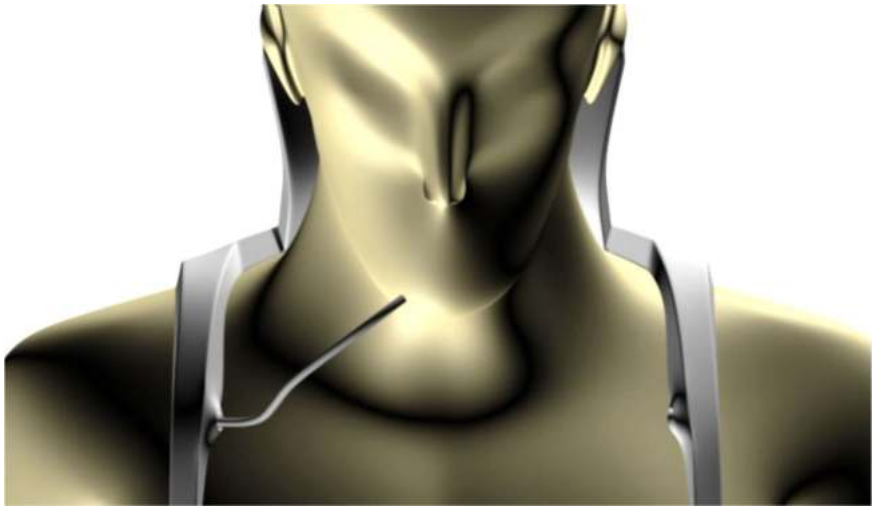

Figura 27: Ubicación de la manguera médica

El **RX** dispone de un contenedor, también diseñado con el software **3D Studio,** que se muestra en la *Figura 28.*

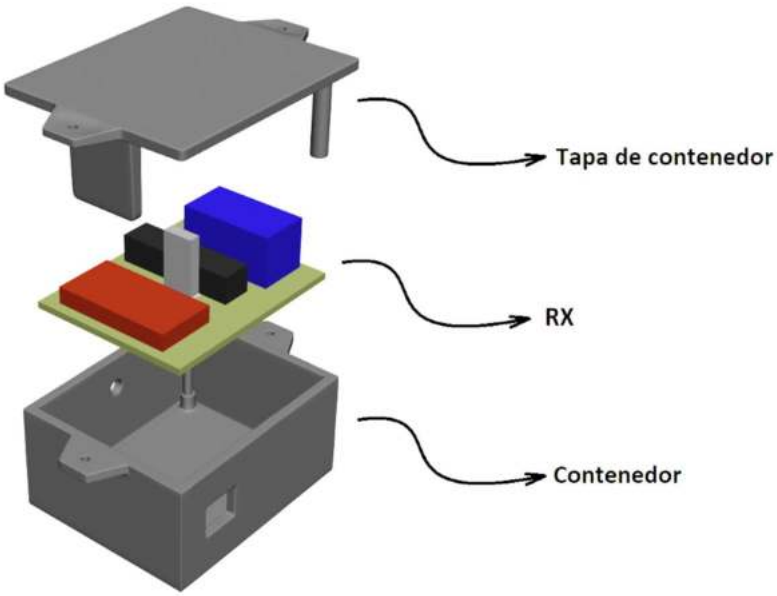

*Figura 28: Contenedor de **RX.***

El contenedor posee una abertura para conectar el cable USB y también cuenta con un Led que indica el funcionamiento del sistema. A partir del uso de una impresora 3D se llevan a cabo los diseños realizados.

Con el fin de captar de forma correcta los movimientos del puntero se diseña una vincha que se aloja en la parte posterior de la cabeza del usuario. De esta forma el sensor infrarrojo puede funcionar de manera óptima, permitiendo mayor fluidez en el control del cursor.

7. Resultados

Al final del proyecto **InterMOUSE** se obtuvo un dispositivo capaz de suplir las limitaciones de un individuo en condición de cuadriplejia, para que el mismo pueda comunicarse de forma eficaz con un sistema operativo alojado en una PC. El dispositivo posee características inalámbricas, es adaptable físicamente a casi cualquier usuario que lo requiera, es de dimensiones acotadas y sencillo de utilizar. El sistema requiere solo de una alimentación a través de una batería de 9v. A su vez, **InterMOUSE** posee un diseño atractivo a la vista y de gran practicidad en su utilización diaria, intentando minimizar la asistencia al usuario por parte de un tercero. El usuario solo necesitará asistencia para colocarse y ajustar el soporte **SIM,** y para la conexión por única vez del receptor. En el caso de que el usuario lo desee, el sistema **InterMOUSE** con su soporte **SIM** puede quedar ajustado al mismo mientras este realiza otra actividad, sin presentar una molestia. A través del **CIIM,** el sistema operativo de la PC ya inicia con la interfaz gráfica de **InterMOUSE** lista para usar, por lo tanto, tampoco se necesita asistencia en este caso.

8. Discusión

Más allá de que se han logrado cumplir los objetivos principales del proyecto, caben mencionar ciertos detalles que podrían mejorar el desempeño del producto. Uno de ellos se enfoca en la reducción del consumo del dispositivo, enviando datos sólo cuando el usuario ejecute una acción, y no en intervalos de tiempo regulares. En este caso se pretende alimentar el **TX** con la batería que se encuentra en las sillas de rueda eléctricas, es por eso que no se considera que este consumo sea significativo, en su defecto, también se puede alimentar al **TX** con una batería de 9v externa. Por otro lado, el **RX** se encuentra alimentado por el puerto USB de la PC.

En cuanto al **SIM**, una posible mejora consiste en ubicar el transmisor en la parte frontal, en vez de posterior. Esto no solo permitiría un mejor acceso a la manguera médica, sino que además proporcionaría un mejor balance del peso al **SIM**.

9. Conclusión

La misión principal de este proyecto consistía en complementar las limitaciones provocadas por una lesión de cuadriplejia, de forma que le permita a un usuario en esta condición, acceder de una manera sencilla a las funcionalidades propias del sistema operativo de una PC. De esta forma, se puso como objetivo mejorar la calidad de vida del usuario de **InterMOUSE,** proporcionándole una herramienta de trabajo para desarrollarse en las actividades que desee. La estética y practicidad del sistema fueron igual de cuidadas que la funcionalidad del mismo, obteniendo un producto duradero, y de gran calidad. El sistema **InterMOUSE** ha sido probado tanto por sus diseñadores como por usuarios en la condición de cuadriplejia y el mismo no provoca ninguna molestia, incluso luego de largas horas de uso. La curva de adaptación al sistema es buena, incluso a los pocos minutos ya se nota un avance y un uso más rápido de las funciones. Se observa que, a las pocas horas de utilización del sistema, el usuario lo maneja con naturalidad.

Luego de un periodo de prueba y error sobre cuál sería la mejor secuencia de soplidos y succiones para controlar las funcionalidades de **InterMOUSE**, se llega a un buen resultado, teniendo en cuenta que el objetivo era que el uso fuese lo más intuitivo posible.

La interfaz gráfica **CII** provee una adecuada forma de configuración del sistema, y lo más importante es que el usuario lo puede realizar solo todas las veces que quiera, sin pedir asistencia a un tercero.

Como se dijo previamente, tanto la estética del producto como la funcionalidad eran prioridad en este proyecto, y esto se demuestra con la incorporación del soporte **SIM.** El **SIM** fue diseñado para las necesidades específicas de este dispositivo y fue impreso en una impresora 3D, integrándose perfectamente al sistema **InterMOUSE.**

Luego de visualizar el sistema en funcionamiento, se hace más evidente lo importante que resulta la característica inalámbrica, ya que se vuelve una prioridad para la practicidad de uso del mismo.

A partir de lo expresado anteriormente, se concluye que los resultados del proyecto han sido satisfactorios, ya que se han logrado cumplir los objetivos establecidos al inicio del mismo.