INSTITUTO TECNÓLOGICO DE TUXTLA GUTIÉRREZ

Ingeniería Electrónica

Reporte Final de Residencia Profesional

"Sistema de Identificación de Transporte (SIT)"

Laboratorio de Ingeniería Electrónica

ITTG

Por: Ordóñez Ruíz Luis Fernando
Martínez Carrillo Julio C.

Asesor: M.C. Raúl Moreno Rincón

Tuxtla Gutiérrez, Chiapas Junio 2014

CAPITULO I INTRODUCCION

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados, la mecánica o la informática.

Algunos estaremos confundidos al creer que la robótica podría tener una influencia en este caso, pero hay que aclarar que la robótica no solo es el diseño y construcción de maquinas humanoides o de dispositivos móviles capaces de viajar a Marte o algún otro planeta. La robótica combina diversas disciplinas como son: la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física.

En la actualidad la robótica tiene gran influencia en muchas ramas, y una de ellas son las telecomunicaciones, ahora, es tan común ver sistemas de comunicación inalámbrica o vía remota ya sea a través de computadoras o bien mediante móviles para la transmisión de datos, y es mediante radio frecuencias como esto es posible, es ahí, donde enfocaremos nuestra atención ya que hay un sinfín de maneras de transmitir datos a través de señales, ya sea por Bluethooth, X-bee, MAX232 etc. Es así como llegamos a la mención del modulo nRF24L01 la versión de corto alcance.

A través de ellos la ciencia y la tecnología van avanzando en la optimización de procesos y el envió de información de puntos remotos mediante redes o nodos de radio frecuencia, haciendo este sistema una posibilidad de progreso e innovación en el siglo XXI.

1.1 JUSTIFICACION

El Sistema de Identificación de Transporte (SIT), es una red de módulos de radiofrecuencia con los cuales el usuario podrá utilizar el transporte público con la completa seguridad de que se está subiendo a la ruta que él desea.

Este sistema es solo un pilar más en cuanto al avance tecnológico que ha realizado la Universidad de Reading, en aplicaciones para sistemas de RF, dirigidas especialmente a personas con discapacidad visual, a esta misma discapacidad va dirigido el SIT.

En el Sistema de Identificación de Transporte (SIT), lo que nosotros llamamos “Control” puede ser utilizado por el usuario en cualquier parte de la ciudad por donde transite una ruta de transporte público, en este caso la de su preferencia, ya que el “esclavo” (dispositivo que lleva el colectivo), va ubicado en la parte frontal de la misma, a diferencia del sistema utilizado por la Universidad de Reading en donde las etiquetas RFID que utiliza están estáticas en cada esquina de la ciudad, y con un alcance de 1 metro, si tomamos en cuenta el alcance, el módulo nRF24L01 utilizado por el SIT tendrán un alcance de entre 20 – 30 mts. Dependiendo del flujo de interferencia en el medio con respecto al usuario, dándole tiempo de reacción al mismo para poder abordar con cuidado.

El SIT surge con la finalidad de resolver el problema tan grande que tienen las personas con discapacidad visual para utilizar el transporte público, ya que al no tener esa seguridad y confianza recurren al apoyo de otros usuarios del transporte causando molestias y en algunos casos son ignorados, por ello este sistema pretende lidiar con esta problemática dándole esa autonomía que necesita el invidente. Cabe mencionar que no es un sistema cerrado a un grupo social, que en este caso son las personas invidentes, si no que cualquier tipo de persona y en diferentes contextos podría hacer uso de este sistema.

Además el transporte público existe en todo el mundo, por ello, este sistema puede ser utilizado de la misma forma y con la misma eficiencia en cualquier parte

del mundo, haciéndole las modificaciones que sean necesarias para adaptarlo al medio.

Por otra parte la implementación de este sistema conlleva a que se incremente el número de usuarios del transporte público, ya que actualmente los invidentes recurren a alguien que los auxilie en actividades de este tipo, al brindarles esa seguridad ellos pueden hacer uso del transporte de manera autónoma incrementando así a mediano plazo el número de usuarios y los ingresos en cada ruta de colectivo donde se implemente el sistema. En una economía en crecimiento como lo es la del país en el que vivimos, un proyecto visionario de esta magnitud podría desembocar en un despunte económico para el servicio de transporte público.

Como tal el SIT no presenta un impacto ambiental ya que no produce ningún tipo de contaminantes en cualquiera de sus versiones, por el contrario el material que se implementará es reutilizable.

El desarrollo del sistema es un pilar mas en cuanto al avance tecnológico para aplicaciones con RF, ya que tanto el campo de investigación para discapacidades y la aplicación de este tipo de módulos es un campo muy amplio que necesita de ideas nuevas y con un impacto importante como es el caso del SIT, ya que hay que destacar que este sistema es completamente innovador y versátil, puesto que se puede manipular y queda abierto a nuevas ideas y aportaciones para enriquecer su funcionamiento. Hay muchas dudas y problemas por resolver y el SIT aportará una pequeña solución a uno de los muchos problemas con los que lidian día a día las personas con discapacidad visual.

1.2 OBJETIVOS

1.2.1 Objetivo General:

Construir e implementar un sistema de identificación de rutas de colectivo a partir de módulos de radiofrecuencia de mediano-largo alcance.

1.2.2 Objetivos Específicos:

- Diseño y modelado de dispositivos (maestro – esclavo).
- Desarrollo de códigos y protocolos de comunicación.
- Armado y acoplamiento de subsistemas.
- Prueba del Sistema en el medio y corrección.
- Prueba final y reporte.

1.3 PROBLEMAS A RESOLVER

Los problemas a resolver son los siguientes:

- Diseño y modelado de dispositivos (maestro – esclavo).
- Desarrollo de códigos y protocolos de comunicación.
- Armado y acoplamiento de subsistemas.
- Prueba del Sistema en el medio y corrección.
- Prueba final y reporte.

1.4 ALCANCES Y LIMITACIONES

1.4.1 Alcances

Puesto que el pilar de nuestro proyecto y prototipo es el modulo nRF24l01, mencionaremos los alcances que tiene:

- Alcance de envió y recepción de hasta 80 metros reales.
- Poco consumo de corriente.
- Al ser un transceptor, ofrece la posibilidad de crear una red de nodos para comunicaciones full-dúplex.
- Maneja hasta dos velocidades de transmisión 1Mbps y 2Mbps.

1.4.2 Limitaciones

Ningún sistema electrónico puede ser 100% perfecto y como tal tiene ciertas limitaciones, así pues el nRF24L01 también.

- A campo abierto y con edificios a la redonda de 80 mts, la señal puede atenuarse considerablemente a un 50%.
- Durante el proceso pueden presentarse obstrucciones en la transmisión.
- No contar con recursos suficientes para mejorar la comunicación.
- Al ser una señal tipo wireless, el envío es en forma de radio, lo que presentaría un problema al cruzar señales.

CAPITULO II CARACTERIZACION DEL AREA

2.1 Antecedentes de la empresa

En la década de los 70s, se incorpora el estado de Chiapas al movimiento educativo nacional extensión educativa, por intervención del Gobierno del Estado de Chiapas ante la federación.

Esta gestión dio origen a la creación del Instituto Tecnológico Regional de Tuxtla Gutiérrez (ITRTG) hoy Instituto Tecnológico de Tuxtla Gutiérrez (ITTG). El día 23 de agosto de 1971 el Gobernador del Estado, Dr. Manuel Velasco Suárez, colocó la primera piedra de lo que muy pronto sería el Centro Educativo de nivel medio superior más importante de la entidad. El día 22 de octubre de 1972, con una infraestructura de 2 edificios con 8 aulas, 2 laboratorios y un edificio para talleres abre sus puertas el Instituto Tecnológico de Tuxtla Gutiérrez con las carreras de Técnico en Máquinas de Combustión Interna, Electricidad, Laboratorista Químico y Máquinas y Herramientas.

En el año 1974 dio inicio la modalidad en el nivel superior, ofreciendo las carreras de Ingeniería Industrial en Producción y Bioquímica en Productos Naturales.

En 1980 se amplió la oferta educativa al incorporarse las carreras de Ingeniería Industrial Eléctrica e Ingeniería Industrial Química.

En 1987 se abre la carrera de Ingeniería en Electrónica y se liquidan en 1989 las carreras del sistema abierto del nivel medio superior y en el nivel superior se reorientó la oferta en la carrera de Ingeniería Industrial Eléctrica y se inicia también Ingeniería Mecánica.
En 1991 surge la licenciatura en Ingeniería en Sistemas Computacionales. Desde 1997 el Instituto Tecnológico de Tuxtla Gutiérrez ofrece la Especialización en Ingeniería Ambiental como primer programa de postgrado.

En 1998 se estableció el programa interinstitucional de postgrado con la Universidad Autónoma de Chiapas para impartir en el Instituto Tecnológico la Maestría en Biotecnología.

En el año 1999 se inició el programa de Maestría en Administración como respuesta a la demanda del sector industrial y de servicios de la región. A partir de 2000 se abrió también la Especialización en Biotecnología Vegetal y un año después dio inicio el programa de Maestría en Ciencias en Ingeniería Bioquímica y la Licenciatura en Informática.

El instituto tecnológico de Tuxtla Gutiérrez está ubicado en la carretera panamericana Km. 1080 C.P 29050 como se puede observar en la *Figura 2.1*

Figura 2.1 Ubicación del Instituto Tecnológico de Tuxtla Gutiérrez

2.1.1 Misión, visión y valores

Misión

Formar de manera integral profesionistas de excelencia en el campo de la ciencia y la tecnología con actitud emprendedora respeto al medio ambiente y apego a los valores éticos.

Visión

Ser una institución de excelencia en la educación superior tecnológica del sureste comprometido con el desarrollo socioeconómico sustentable de la región.

Valores

- El ser humano
- El espíritu de servicio
- El liderazgo
- El trabajo en equipo
- La calidad
- El alto desempeño
- Respeto al medio ambiente

2.2 Descripción del área de trabajo

Dentro del ITTG existen un área específica para la carrera de Electrónica, dicha área es el laboratorio de Electrónica en el edificio I, en este se encuentra el área de robótica, laboratorios de electrónica digital y analógica, además del laboratorio donde se imparte la clase de Controladores Lógicos Programable (PLC), es en este edificio en donde se desarrollo el trabajo de residencia, puesto que se contaba con un área idónea para el trabajo.

Dentro del material utilizado esta:

- PC de trabajo.
- Multímetro.
- Fuentes conmutadas.

CAPITULO III FUNDAMENTOS TEORICOS

3.1 Comunicación Inalámbrica

Nuestra naturaleza humana nos hace desenvolvernos en situaciones donde se requiere comunicación. Para ello, es necesario establecer medios para que esto se pueda realizar. Uno de los medios más discutidos es la capacidad de comunicar computadores o cualquier otro dispositivo a través de redes inalámbricas.

La comunicación inalámbrica o sin cables es aquella en la que el emisor y el receptor no se encuentran unidos por un medio de propagación físico, sino que se utiliza la modulación de ondas electromagnéticas a través del espacio. En este sentido, los dispositivos físicos sólo están presentes en los emisores y receptores de la señal, entre los cuales encontramos: antenas, computadoras portátiles, PDA, teléfonos móviles, etc.

Figura 3.1 Ilustración comunicación inalámbrica

La comunicación inalámbrica, que se realiza a través de ondas de radiofrecuencia, facilita la operación en lugares donde el dispositivo o un PC no se encuentran en una ubicación fija (almacenes, oficinas de varios pisos, etc.). Actualmente se utiliza de una manera general y accesible para todo público. Cabe también mencionar que actualmente las redes cableadas presentan ventaja en cuanto a transmisión de datos sobre las inalámbricas. Mientras que las cableadas proporcionan velocidades de hasta 1 Gbps (Red Gigabit), las inalámbricas alcanzan sólo hasta 108 Mbps.

A partir de esta definición podemos clasificar el tipo de transmisión de acurdo a la frecuencia a la que estamos trabajando como se muestra en la siguiente imagen.

Tabla 3.1 Tabla de relación de bandas de trasmisión

Nombre	Abreviatura inglesa	Banda ITU	Frecuencias	Longitud de onda
			Inferior a 3 Hz	>100.000 km
Extra baja frecuencia / *Extremely low frequency*	ELF	1	3-30 Hz	100.000-10.000 km
Super baja frecuencia / *Super low frequency*	SLF	2	30-300 Hz	10.000-1.000 km
Ultra baja frecuencia / *Ultra low frequency*	ULF	3	300-3000 Hz	1.000-100 km
Muy baja frecuencia / *Very low frequency*	VLF	4	3-30 KHz	100-10 km
Baja frecuencia / *Low frequency*	LF	5	30-300 KHz	10-1 km
Media frecuencia/ *Medium frequency*	MF	6	300-3.000 KHz	1 km-100 m
Alta frecuencia / *High frequency*	HF	7	3-30 MHz	100-10 m
Muy alta frecuencia / *Very high frequency*	VHF	8	30-300 MHz	10-1 m
Ultra alta frecuencia / *Ultra high frequency*	UHF	9	300-3.000 MHz	1 m-100 mm
Super alta frecuencia / *Super high frequency*	SHF	10	3-30 GHz	100-10 mm
Extra alta frecuencia / *Extremely high frequency*	EHF	11	30-300 GHz	10-1 mm
			Por encima de los 300 GHz	<1 mm

Gracias a esta clasificación es como sabemos que la frecuencia de 2.4GHz con la que trabaja el modulo transceptor está ubicada en la banda de UHF, a continuación explicaremos las características de este modulo.

3.2 Transceptor Nórdico nRF24L01

El transceptor nórdico nRF24L01 tiene un alto grado de integración, y es considerado del tipo ULP ("Ultra bajo consumo" por sus siglas en ingles), es un transceptor que trabaja con hasta 2Mbps en la banda de los 2.4 GHz ISM (Industrial, Scientific and Medical).

A continuación se muestran algunas de sus características:

Características:

El Nordic Semiconductor nRF24L01 desciende del TXRX24G, pero ofrece algunas características importantes que hacen que sea mucho más fácil de tratar:

- Mejorado ShockBurst ™: Este es un protocolo de capa de enlace de los países nórdicos que le da al auto-reconocimiento de la radio, auto-retransmitir, y la capacidad de detección de pérdida de paquetes. Esto hará que el enlace inalámbrico mucho más fiable sin añadir complejidad a los programas de aplicación de los estudiantes.
- Bus SPI: El nRF24L01 implementa un 4 hilos de serie bus Peripheral Interface. Esto permite que el controlador de software para utilizar el módulo SPI alta velocidad nativa del microcontrolador para las comunicaciones en lugar de tener que hacer bashing poco manual, que es menos confiable, más lenta y más difícil de implementar.
- Antena externa: Esto debería mejorar la sensibilidad de la nueva radio y rango sobre los de las viejas radios.
- MultiCeiver ™: Cada radio puede recibir paquetes en un máximo de seis direcciones diferentes. Esto nos permite implementar características tales como la radiodifusión paquete selectiva sin renunciar a otras funciones.

Otras características: ancho variable de paquetes, gestión de colas de paquetes, ack carga útil de paquetes, etc.

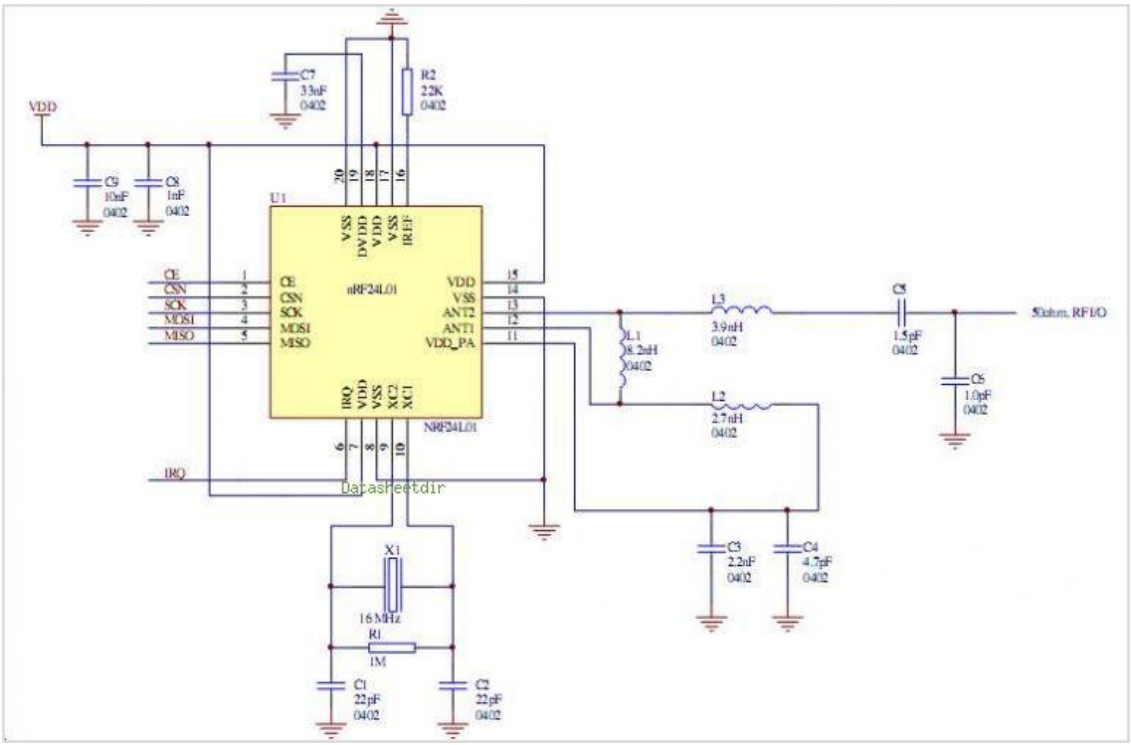


Figura 3.2 Esquema del nRF24L01

El módulo trabaja con 8 pines principales, seis de ellos son esenciales para la comunicación entre módulos, ya que trabaja con el protocolo SPI, y los otros dos pines libres son masa (-) y alimentación (+), es recomendable para un mejor funcionamiento de los módulos, colocar un capacitor de entre 10 y 100uF entre masa y alimentación para evitar lo que se conoce como voltaje de rizo.

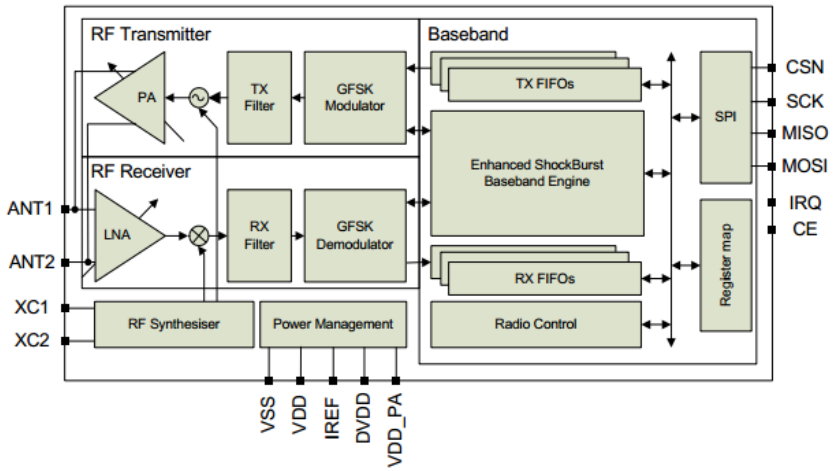


Figura 3.3 Composición del CI nRF24L01

El Bus SPI (del inglés *Serial Peripheral Interface*) que mencionamos anteriormente es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj.

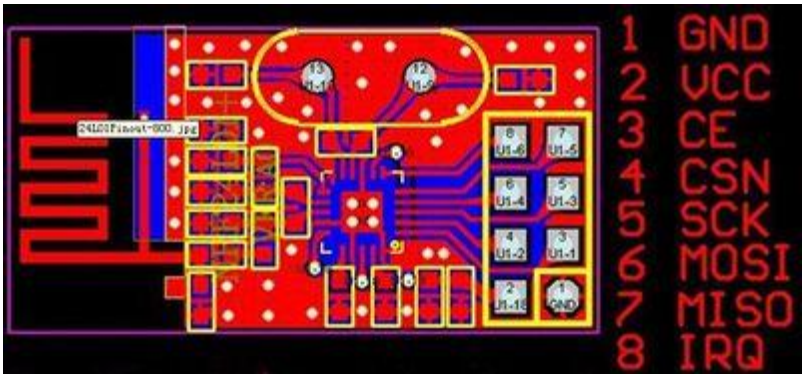


Figura 3.4 Pines de interfaz SPI

- CE. Chip de habilitación, activa el modo Tx o Rx.
- CSN. Selecciona el chip SPI.
- SCK. Reloj del protocolo SPI.
- MOSI. Entrada de datos SPI esclavo.
- MISO. Salida de datos SPI esclavo, con opción de tri-estado.
- IRQ. Pin de interrupción, se activa en estado bajo.
- VDD. Alimentación (+1.9V - +3.6V DC).
- VSS. Tierra (0V).

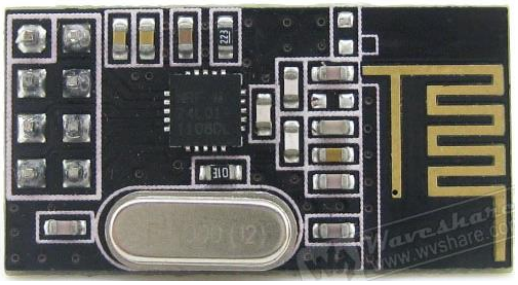


Figura 3.5 Modulo nRF24L01

3.3 Plataforma ARDUINO

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).

3.3.1 Arduino UNO

El Arduino Uno es una placa electrónica basada en el microprocesador Atmega328 (ficha técnica). Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 pueden ser utilizados como salidas PWM), 6 entradas analógicas, un resonador cerámico 16 MHz, una conexión USB, un conector de alimentación, un header ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador, simplemente conectarlo a un PC con un cable USB, o alimentarla con un adaptador de corriente AC a DC para empezar.

El Arduino Uno se diferencia de todas las placas anteriores en que no utiliza el chip controlador de USB a serial FTDI. En lugar de ello, se cuenta con el Atmega16U2 (Atmega8U2 hasta la versión R2) programado como convertidor USB a serie.

El Arduino Uno puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente.

En la alimentación externa (no USB) de potencia puede venir con un adaptador de AC-DC ó la batería.
La tarjeta puede funcionar con un suministro externo de 6 a 20 voltios. Si se proporcionan menos de 7V, no obstante, el pin de 5V puede suministrar menos de cinco voltios y el circuito puede ser inestable. Si se utiliza más de 12V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son como sigue:

- VIN. La tensión de entrada a la placa Arduino cuando se trata de utilizar una fuente de alimentación externa (en contraposición a 5 voltios de la conexión USB o de otra fuente de alimentación regulada). Usted puede suministrar tensión a través de este pin, o, si el suministro de tensión a través de la toma de poder, acceder a ella a través de este pin.
- 5V. Este pin como salida una 5V regulado por el regulador en el tablero. El tablero puede ser alimentado ya sea desde el conector de alimentación de CC (7 - 12), el conector USB (5V) o el pasador de VIN del tablero (7-12V). El suministro de tensión a través de los 5V o 3.3V pins no pasa por el regulador, y puede dañar su tablero. No aconsejamos ella.
- 3V3. Un suministro de 3,3 voltios generados por el regulador a bordo. El drenaje actual máximo es de 50 mA.
- GND. patillas de tierra.
- IOREF. Este pin de la placa Arduino proporciona la referencia de tensión con la que opera el microcontrolador. Un escudo configurado puede leer el voltaje pin IOREF y seleccione la fuente de alimentación adecuada o habilitar traductores tensión en las salidas para trabajar con los 5V o 3.3V.

Memoria

El ATmega328 tiene 32 KB (con 0,5 KB utilizado por el gestor de arranque). También dispone de 2 KB de SRAM y 1 KB de EEPROM.

Entradas y salidas

Cada uno de los 14 pines digitales en el Arduino Uno se puede utilizar como una entrada o salida, utilizando pinMode (), digitalWrite (), y digitalRead () funciones. Operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA. Y tiene una resistencia de pull-up (desconectado por defecto) de 20-50 ohm. Además, algunos pines tienes funciones especializadas como las siguientes:

- De serie: 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y de transmisión (TX) TTL datos en serie. Estos pines están conectados a los pines correspondientes del ATmega8U2 USB-to-TTL de chips de serie.
- Interrupciones externas. 2 y 3 Estos pines pueden ser configurados para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor. Ver el attachInterrupt () función para más detalles.
- PWM: 3, 5, 6, 9, 10, y 11 proporcionan PWM de 8 bits con el () analogWrite función.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) Estos pines soportan comunicación SPI utilizando la biblioteca de SPI .
- LED: 13 Hay un built-in LED conectado al pin digital 13. Cuando el pin es de alto valor, el LED está encendido, cuando el pasador es bajo, es apagado.

El Arduino Uno tiene 6 entradas analógicas, etiquetados A0 a A5, cada uno de los cuales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto se miden desde el suelo a 5 voltios, aunque es posible cambiar el extremo superior de su rango utilizando el pin A_{REF} y la analogReference () función. Además, algunos pernos han especializado funcionalidad:

- TWI: A4 o A5 pin SDA y SCL o pin. Apoyar la comunicación TWI utilizando la librería Wire .

Hay un par de pines la placa:

- AREF. Voltaje de referencia para las entradas analógicas. Se utiliza con analogReference ().
- Restablecer. Con esta línea al pin BAJO para reiniciar el microcontrolador. Normalmente se utiliza para agregar un botón de reinicio para escudos que bloquean el Arduino uno en el tablero.

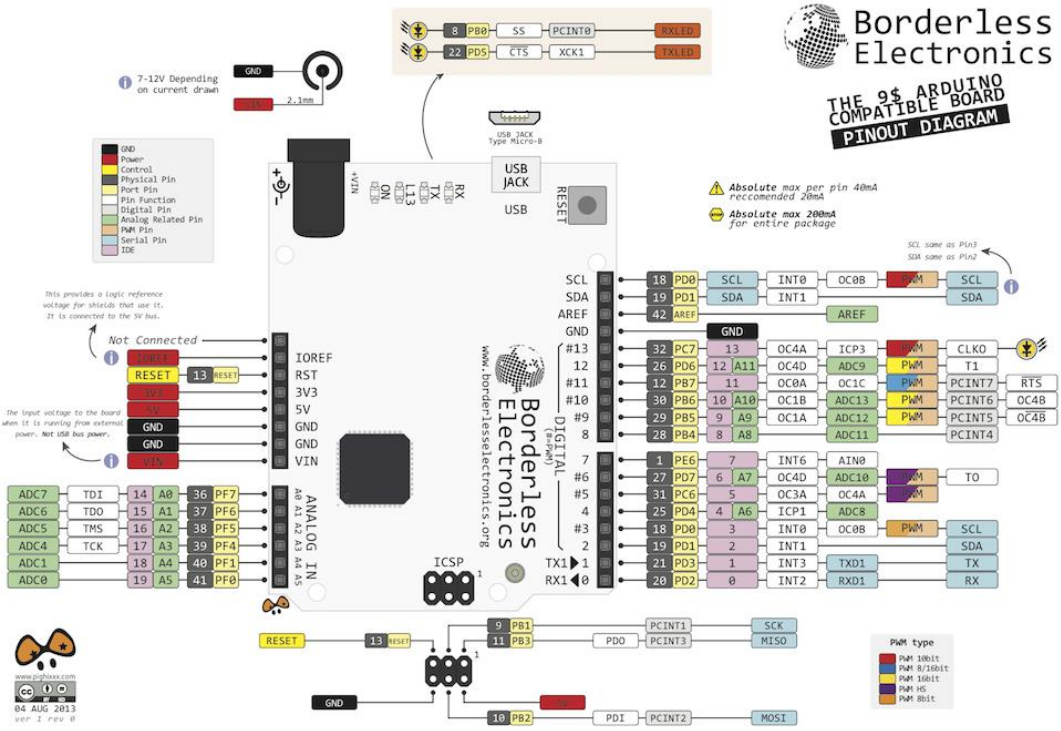


Figura 3.6 Descripción de Arduino UNO

3.3.2 Arduino MEGA

El Arduino Mega es una placa microcontrolador basada ATmeg1280. Tiene 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset. Contiene todo lo necesario para hacer funcionar el microcontrolador; simplemente conéctalo al ordenador con el cable USB o aliméntalo con un trasformador o batería para empezar. El Mega es compatible con la mayoría de shields diseñados para el Arduino Duemilanove o Diecimila.

Alimentación

El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.
Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería pueden conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER)
La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan mas de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- VIN. La entrada de voltaje a la placa Arduino cando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede

proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.

- 5V. La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
- 3.3V. Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.
- GND. Pines de toma de tierra.

Memoria

El ATmega1280 tiene 128KB de memoria flash para almacenar código (4KB son usados para el arranque del sistema (bootloader). El ATmega1280 tiene 8 KB de memoria SRAM. El ATmega1280 tiene 4KB de EEPROM, que puede a la cual se puede acceder para leer o escribir con la librería EEPROM.

Entradas y Salidas

Cada uno de los 54 pines digitales en el Duemilanove pueden utilizarse como entradas o como salidas usando las funciones pinMode (), digitalWrite (), y digitalRead () . Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad máxima de 40mA y tiene una resistencia interna (desconectada por defecto) de 20-50kOhms. Además, algunos pines tienen funciones especializadas:

- Serie: 0 (RX) y 1 (TX), Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Usado para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) están conectados a los pines correspondientes del chip FTDI USB-to-TTL.
- Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2). Estos pines se pueden configurar para lanzar una interrupción en un valor LOW (0V), en flancos

de subida o bajada (cambio de LOW a HIGH (5V) o viceversa), o en cambios de valor.

- PWM: de 0 a 13. Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a través de la función analogWrite ().
- SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK). Estos pines proporcionan comunicación SPI, que a pesar de que el hardware la proporcione actualmente no está incluido en el lenguaje Arduino.
- LED: 13. Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de tierra a 5 voltios, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference(). Además algunos pines tienen funciones especializadas:

- I^2C: 20 (SDA) y 21 (SCL). Soporte del protocolo de comunicaciones I^2C (TWI) usando la librería.

Hay unos otros pines en la placa:

- AREF. Voltaje de referencia para la entradas analógicas.
- Reset. Suministrar un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

Comunicaciones

El Arduino Mega facilita en varios aspectos la comunicación con el ordenador, otro Arduino u otros microcontroladores. El ATmega1280 proporciona cuatro puertos de comunicación vía serie UART TTL (5V). Un chip

FTDI FT232RL integrado en la placa canaliza esta comunicación serie a traes del USB y los drivers FTDI (incluidos en el software de Arduino) proporcionan un puerto serie virtual en el ordenador. El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDS RX y TX de la placa parpadearan cuando se detecte comunicación transmitida través del chip FTDI y la conexión USB (no parpadearan si se usa la comunicación serie a través de los pines 0 y 1).

La librería Software Serial permite comunicación serie por cualquier par de pines digitales del Mega.

El ATmega1280 también soporta la comunicación I2C (TWI) y SPI. El software de Arduino incluye una librería Wire para simplificar el uso el bus I2C.

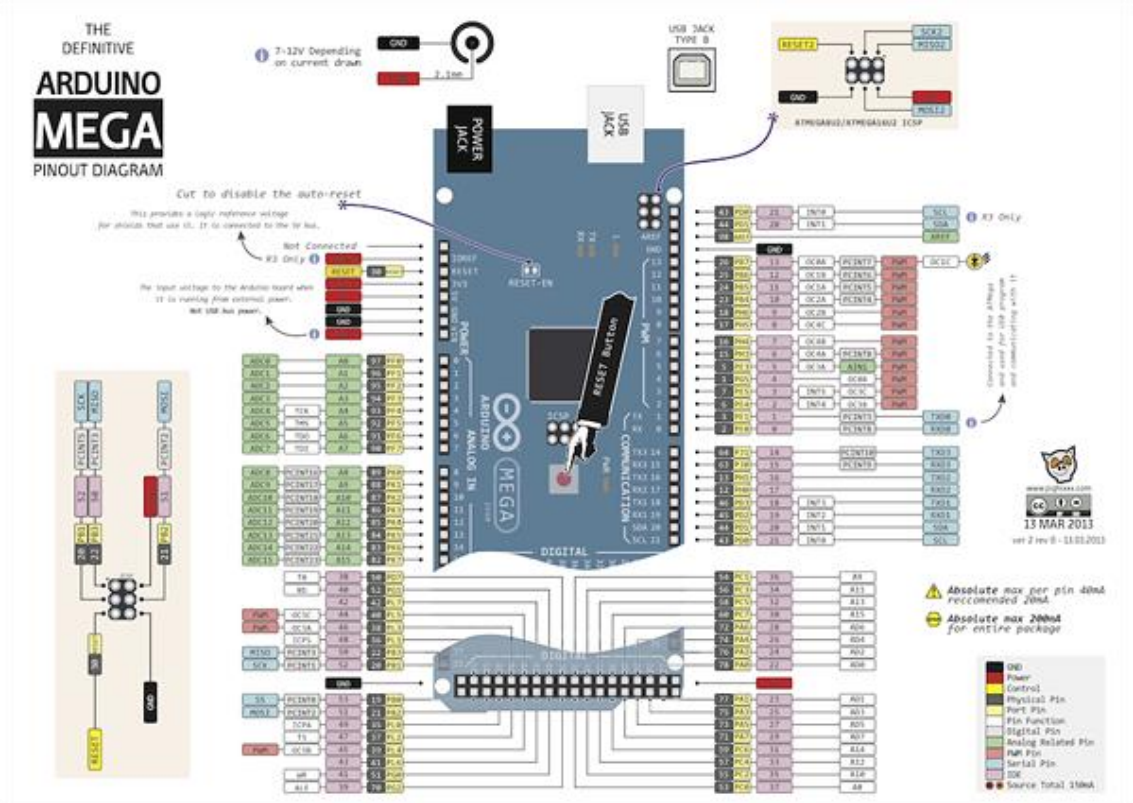


Figura 3.7 Descripción Arduino MEGA

3.4 Teclado matricial

El teclado matricial como su nombre lo indica, esta formado por una matriz de botones, que en este caso será de 3x4, tres columnas por cuatro filas, cabe mencionar que cuenta con su propia librería dentro del simulador de arduino, la cual elimina la necesidad de utilizar resistencias y diodos, puesto que la librería utiliza las resistencias pull up para eliminar las resistencias normales.

A diferencia del teclado convencional, este es de un platico rígido para “uso rudo” como coloquialmente se le dice, permitiendo soportar mayor presión sobre sus teclas sin que se deforme su figura.

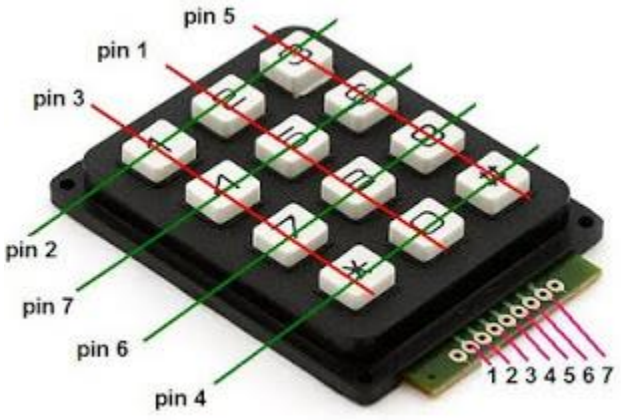


Figura 3.8 Relación de pines del teclado

Figura 3.9 Teclado Sparkfun 3x4

CAPITULO IV DESCRIPCION DE LAS ACTIVIDADES REALIZADAS

Para la elaboración y diseño del sistema se han descrito con anterioridad los componentes que han sido previamente seleccionados por su versatilidad y factibilidad de utilización dentro de una amplia variedad de componentes electrónicos.

A continuación se describen a detalle cada uno pasos que se llevaron a cabo para el desarrollo del proyecto.

4.1 Reconocimiento de modulo nRF24L01 y plataforma Arduino.

Para poder comenzar con el proyecto como tal se debe de conocer la configuración que guarda el modulo con respecto al tipo de placa arduino que utilizaremos, en este caso al trabajar con dos tipos diferentes, se debe conocer su configuración y el tipo de conexión que requiere.

Tabla 4.1 Pines para conectar el modulo a las placas UNO y MEGA de arduino

MODULE PIN	DESCRIPTION	UNO	MEGA
1	GND	GND	GND
2	VCC	3.3V (See note)	3.3V (See note)
3	CE	DIO 8	DIO 8
4	CSN	DIO 7	DIO 7
5	SCK	DIO 13	DIO 52
6	MOSI	DIO 11	DIO 51
7	MISO	DIO 12	DIO 50
8	IRQ		

WWW.HOBBYCOMPONENTS.COM

Como se puede observar tanto el Arduino UNO como el Arduino MEGA tienen una configuración diferente, debido a que la ubicación de los pines para la interfaz SPI es diferente en cada uno de ellos.

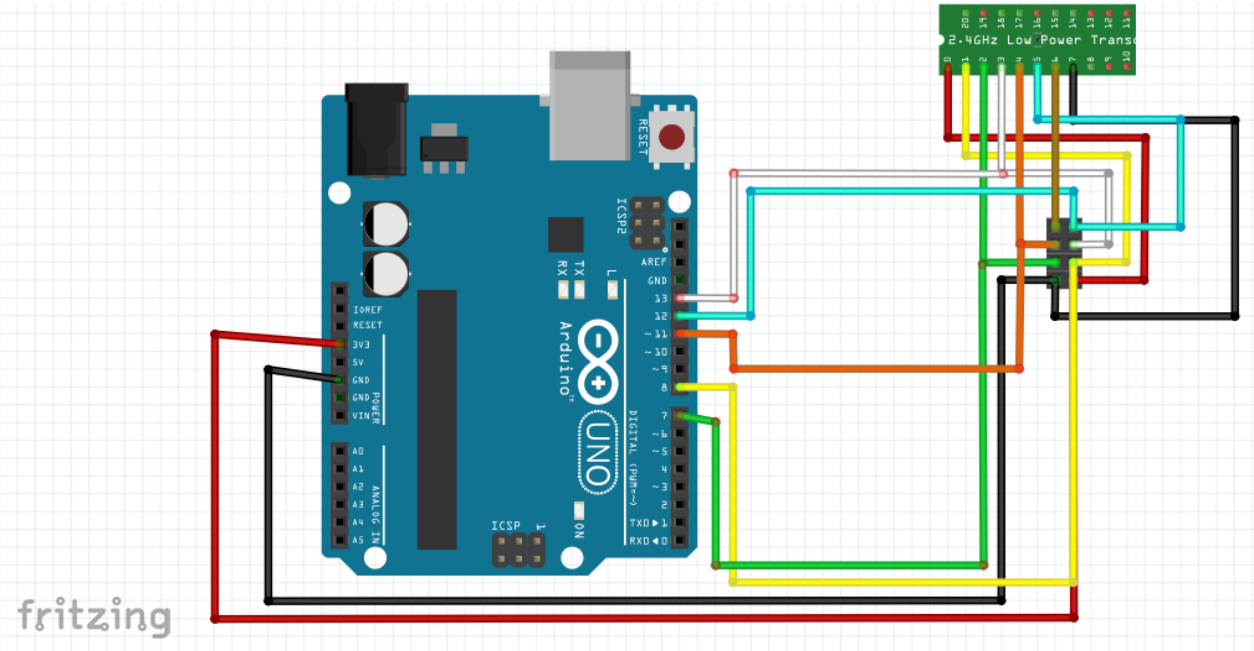


Figura 4.1 Modelado de conexiones Arduino UNO – nRF24L01

Con la ayuda de un software diseñador de circuitos podemos mostrar con líneas de diferente color las conexiones que van de los pines del integrado nRF24L01 hacia 8 pines, simulando como se ve en realidad el modulo y posteriormente sus conexiones directas al Arduino UNO.

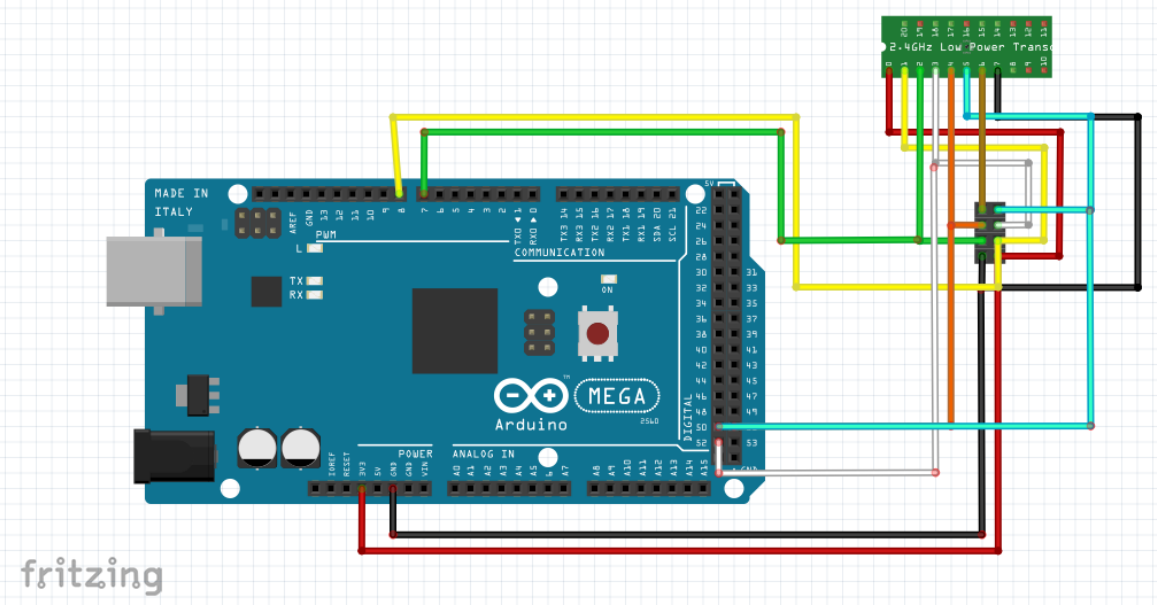


Figura 4.2 Modelado de conexiones Arduino MEGA – nRF24L01

De igual forma podemos observar como es la conexión que debe llevar un arduino MEGA al modulo nRF24L01 para poder establecer una comunicación full-dúplex entre ambos módulos que es lo que buscamos para este proyecto.

4.2 RECONOCIMIENTO DE LAS LIBRERIAS PARA EL nRF24L01

Al igual que cualquier otro dispositivo electrónico externo a la placa de arduino, trabajan con sus propias librerías y dentro de ellas se especifican sus configuraciones, este modulo dependiendo de la versión trabaja con varias, entre ellas destacan <Mirf.h>, <nRF24L01.h>, <RF24.h> y <nRF24L01p.h> (para la versión con antena de este mismo modulo, pero compatible para el que utilizamos en esta ocasión).

Se realizaron pruebas y códigos (debo mencionar que muchas pruebas diferentes) utilizando la librería Mirf.h la cual nos dio un poco de complicaciones,

debido a que no había un ejemplo claro, y en la investigación que se realizo era muy común encontrar los ejemplos puestos dentro del simulador de Arduino y no alguno que facilitara la comprensión de la librería, por ello se decidió cambiar por <nRF24L01p>, librería actualizada para este modulo y con líneas de código y configuración más sencillas y comprensibles, puesto que el 80% de la información obtenida de este modulo era en ingles.

Así pues explicaremos un poco la configuración que presenta esta librería.

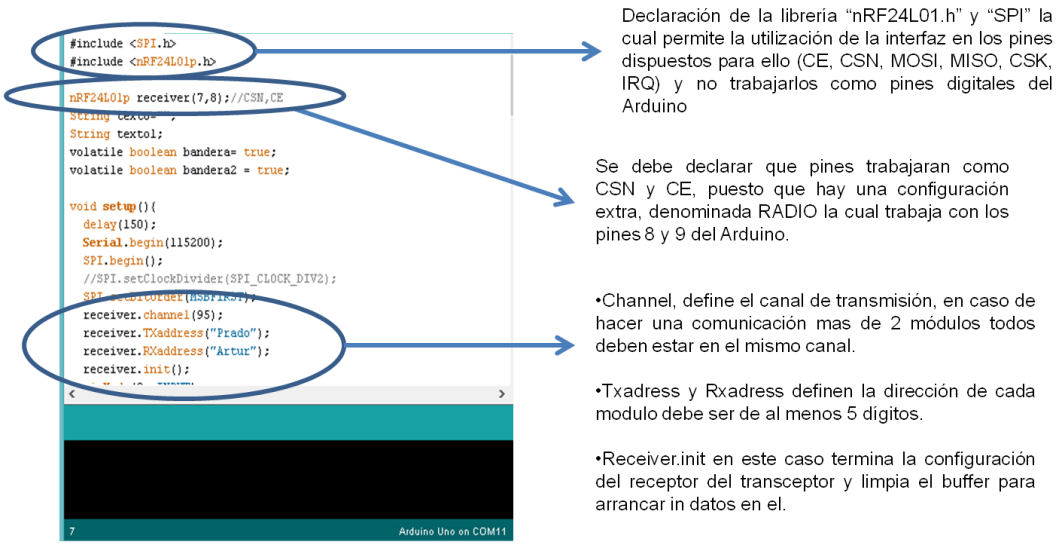


Figura 4.3 Configuración de librería nRF24L01p

4.3 IMPLEMENTACION DE TECLADO NUMERICO AL SISTEMA

Para que el nRF24L01 y el teclado puedan trabajar en conjunto se debe mover un poco la configuración tradicional del teclado puesto que estamos ocupando dos de los siete pines que normalmente son utilizados por la librería <Keypad.h>, la siguiente imagen muestra cómo va conectado el teclado a uno de los arduinos que utilizaremos.

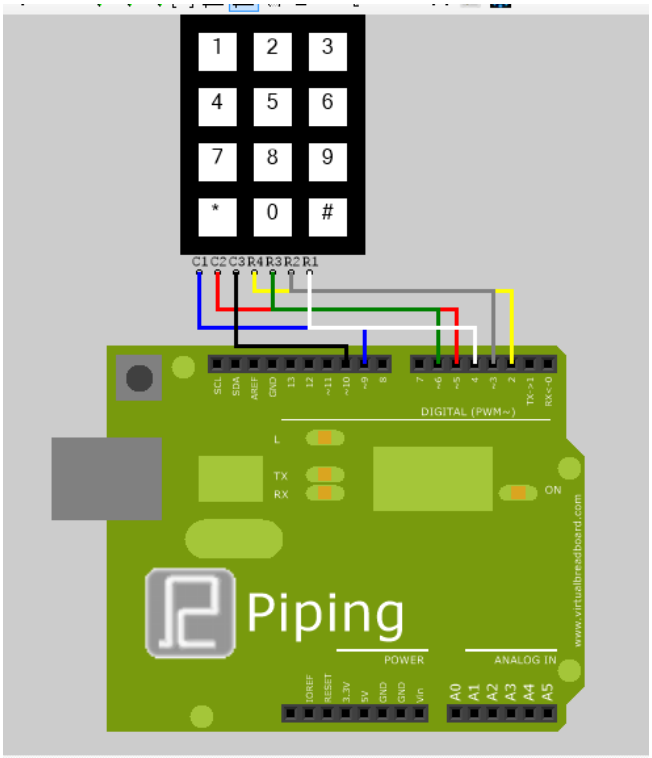


Figura 4.4 Diagrama de conexiones Arduino – Teclado matricial

La imagen muestra lo que asemeja ser un Arduino UNO y un teclado matricial, aunque hay que recalcar que el prototipo final cuenta con un Arduino MEGA, con la misma configuración lo único que varia es la placa de Arduino que se utilizo, debido a que serian necesarios más pines digitales de los que el UNO puede proporcionar.

4.4 ACOMPLAMIENTO DE SUBSISTEMAS PARA EL “CONTROL” (MAESTRO)

Ya que lo que denominamos “control” no solo implica el teclado, se realizo un modelado de las partes secundarias que lo conformarían, para posteriormente hacer el diseño del prototipo físico que portará el usuario.

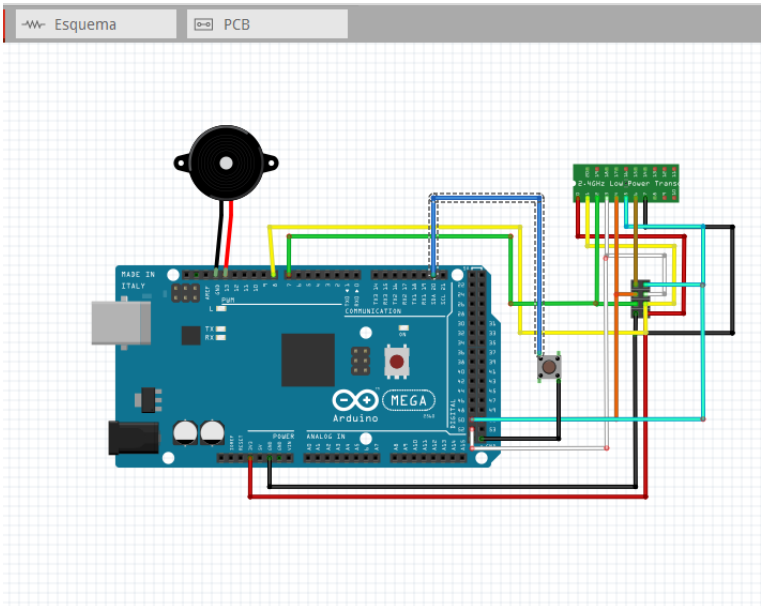


Figura 4.5 Modelado de subsistemas para dispositivo maestro del SIT

El esquema muestra el arduino MEGA el cual es la parte medular del “control”, a su vez lleva conectado un “buzzer” o bocina la cual será la encargada de dar la señal de alerta al usuario en caso de una respuesta tanto positiva como negativa por parte del conductor.

El push bottom que se encuentra conectado al pin 20 digital del arduino corresponde también a uno de los pines que permiten interrupciones externas, con lo cual los procesos se interrumpen para realizar alguna otra acción, en este caso nos servirá como un “Reset”, el cual podrá ser activado bajo las siguientes condiciones:

- Cuando el usuario decida moverse de su posición y ya haya mandado algún numero (en este caso particular el 73) interrumpiendo la transmisión.
- En caso de que hay tecleado la ruta, y después de un rato decida teclear otra en caso de no poder abordar la que deseaba con anterioridad. Interrumpirá la transmisión para reemplazarla por el número que teclee posteriormente.
- En caso de haberse equivocado o no estar seguro del numero que mando (después de ser enviado).

4.5 ACOMPLAMIENTO DE SUBSISTEMAS PARA EL COLECTIVO (ESCLAVO)

El esclavo (denominación utilizada en sistemas de redes de comunicación), no es únicamente el modulo, puesto que se pretende que el colectivo envíe una respuesta al usuario en caso de poder llevarlo o en caso contrario, debe de haber una comunicación bidireccional.

Es por ello que se realizo el modelado de los subsistemas empleados en el dispositivo que portara el colectivo.

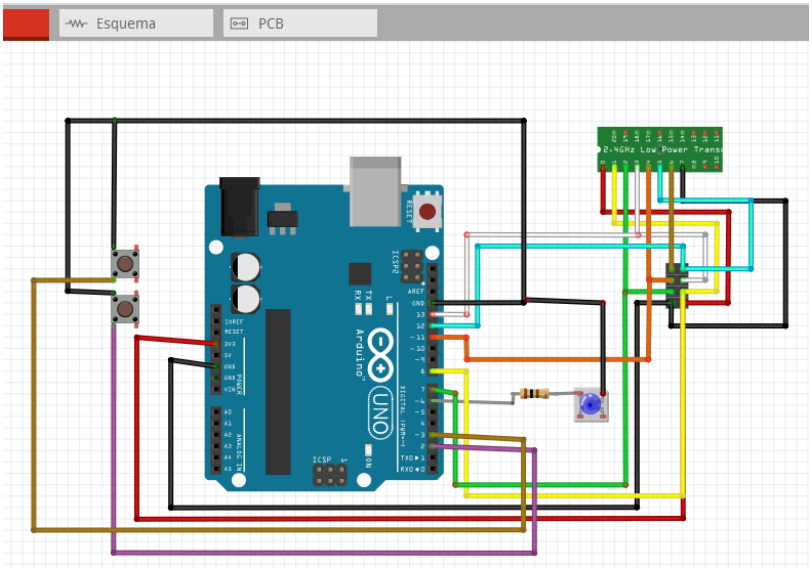


Figura 4.6 Modelado de subsistemas dispositivo esclavo del SIT

Después de haber explicado las conexiones del modulo nRF24L01, pasaremos a mencionar los subsistemas que van anexos a él, el dispositivo esclavo presenta dos pushbottoms del timo gamebottom, el que va conectado al pin digital numero 2, es aquel que permitirá al conductor responder de manera afirmativa y enviar una aprobación al usuario, afirmando que podrá detenerse y brindarle el servicio.

Por contraparte el botón conectado al pin digital 3 envía la negativa.

Al digital numero 6, como se muestra en la imagen va una resistencia de 100 Ohmios, directo a un led ultra brillante color azul de 10 mm, el cual avisará de manera visual al conductor cuando haya algún usuario del SIT en la parada próxima esperando el servicio.

4.6 DESARROLLO DE CODIGOS Y PROTOCOLOS DE COMUNICACIÓN

Es importante mencionar que cualquier comunicación mediante radiofrecuencia debe ser creada bajo un protocolo de comunicación, para no afectar algún otro elemento o sistema que trabaje bajo la misma frecuencia con la que nosotros estamos trabajando, en este caso es la banda de los 2.4GHz. Así pues, el SIT al utilizar un modulo de RF fue configurado al canal 95 (se puede hacer una reconfiguración dependiendo la utilidad o la interferencia que pueda haber en el canal descrito).

Es por ellos que el SIT maneja la siguiente configuración para los dispositivos a utilizar (maestro – esclavo).

4.6.1 Código para dispositivo maestro (“Control”)

Como se puede observar en la figura 4.7 las primeras líneas de código representan la declaración de librerías con las que trabajaremos, mencionamos anteriormente dos de ellas, en esta parte se agregaran la librería <Keypad.h> y <avr/interrupt.h>, Keypad es la encargada de permitirnos utilizar el teclado numérico y asignarle un pin digital a cada pin del teclado, y por otra parte la librería “avr/interrupt” permite el uso de interrupciones externas, en este caso el SIT utiliza una para poder resetear el control y terminar con la transmisión si así lo desea el usuario.

Figura 4.7 Declaración de variables programación del dispositivo maestro

Continuando con la descripción del código, declaramos una serie de variables.

- ***Char key.*** Al presionar la tecla, envía un carácter al arduino el cual lo almacena en esta variable.
- ***String texto.*** Esta variable se utiliza únicamente después de haber concluido con el tecleo de números, puesto que almacena el numero que será transmitido en forma de texto, tomando en cuenta que para que esto

suceda el usuario debe presionar la tecla "*" para poder transmitir el numero que desea.

- ***String texto1.*** Es la variable que recibimos como respuesta del colectivo, la cual puede contener un "Voy" o un "No" en caso contrario la variable estará vacía.
- ***Int x.*** Variable tipo entero, la cual irá incrementando (limitado a 3 incrementos 0-3), hasta que el usuario teclee "*" y así transmitir un numero de una a tres cifras, considerando que en la ciudad de Tuxtla Gutiérrez, Chiapas no hay rutas que transiten dentro de la ciudad con un número mayor a 3 cifras.
- ***Boolean bandera y Boolean bandera2.*** Con ellas se realiza un "switcheo" en las rutinas para intercalar la recepción y la transmisión y así el modulo trabaje de manera eficiente.

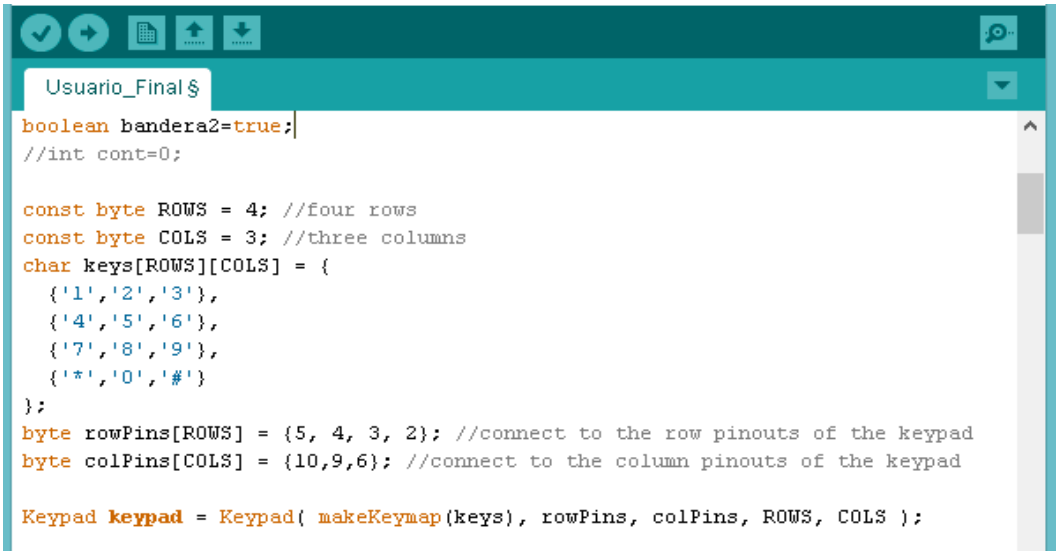


Figura 4.8 Configuración del Teclado con arduino 1.0.5 r2

En la figura 4.8 se muestra la configuración del teclado matricial, aclarando que es un teclado con una matriz de 3x4 (3 columnas y 4 filas), con sus respectivos pines digitales. Nótese que los pines 7 y 8 no son utilizados por el teclado debido a que son utilizados para el modulo de RF.

Usuario_Final §

```
void setup(){
  delay(150);
  Serial.begin(115200);
  SPI.begin();
  //SPI.setClockDivider(SPI_CLOCK_DIV2);
  SPI.setBitOrder(MSBFIRST);
  transmitter.channel(95);
  transmitter.RXaddress("Transporte");
  transmitter.TXaddress("Usuario1");
  transmitter.init();
  pinMode(20, INPUT);
  digitalWrite(20, HIGH);
  attachInterrupt(3,reset,FALLING);
}
```

Figura 4.9 setup () del transmisor del transceptor

Pasamos a la parte del setup (), mencionamos que trabajamos a una velocidad de 115200 Baudios, previa investigación sabemos que el modulo trabaja con mayor eficiencia a esta velocidad.

Definimos las direcciones de transmisión y recepción y a pesar de que este código muestra “transmitter” que nos haría imaginar que es el transmisor únicamente, hay que aclarar que es el transmisor del transceptor, por lo tanto realiza ambas funciones. La función “init ()” limpia el buffer del modulo para enviar que haya una saturación o perdida de información que obstruya el trabajo del modulo o arroje resultados fuera de contexto.

Declaramos el pin digital 20 del arduino MEGA como entrada, ya que es ahí donde colocaremos el botón para la interrupción a través de la función “attachInterrupt”.

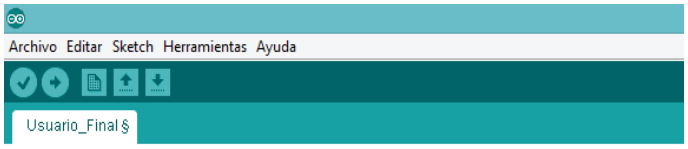
Antes de pasar a la función loop (), hay que mencionar que se trabajó mediante subrutinas así que considero que es importante explicar las subrutinas antes de continuar con el ciclo principal.

A manera de no hacer tedioso y cansado el escrito, explicare brevemente cómo funciona la rutina get (), para la obtención del número de ruta:

En la figura 4.10 podemos ver la función “do” la cual se repetirá en 3 ocasiones siempre y cuando el usuario presione una tecla valida (entiéndase por valido cualquier numero de 0-9 y los caracteres * y #), dicho ciclo se rompe cuando el usuario presiona el botón de asterisco, puesto que está indicando que ya termino de teclear y desea enviar el numero, ya sea de uno, dos o tres cifras, en ese momento en la variable *texto* se carga el valor de las teclas presionadas y activamos el cambio de estado de la variable booleana *bandera* que pasa de falso a verdadero al cumplirse esa condición.

De no presionar asterisco la función *do* se repetirá hasta cumplir las 3 ocasiones y no aceptando un numero mas, en caso de presionar la tecla de numeral (#) el numero almacenado será borrado (por la condición x=0; texto=””;).

Y ahí termina la rutina get ().




```
void get()
  {
   do{
      key= keypad.getKey();}
      while(key==NO_KEY);
     x=x+1;
       switch (x)
         {
           case 1:
           case 2:
           case 3:
           case 4:

   if(key=='*'){bandera=true;}

   else
     {
     if(key=='#') {x=0;texto= "";}
       else texto = texto + key;
     }
   break;
   default: x=0; texto="";
  }
    delay(20);
}
```

Figura 4.10 Rutina get () para obtención del numero de ruta

Pasemos entonces a la rutina transmite (); la cual se cumplirá después de haber presionado la tecla “*” indicando que desea comenzar la transmisión.

```
void transmite()
  {
    transmitter.txPL(texto);
    transmitter.send(SLOW);
    delay(50);
    Serial.println(texto);
    bandera2=false;
}
```

Figura 4.11 Rutina transmite () del transmisor

Como pueden ver la transmisión se hace con pocas líneas de código, recoge la variable *texto* en donde almacenamos el numero, y la transmite. Cabe mencionar que si tomamos el monitor serial del transmisor veremos el número que hemos tecleando.

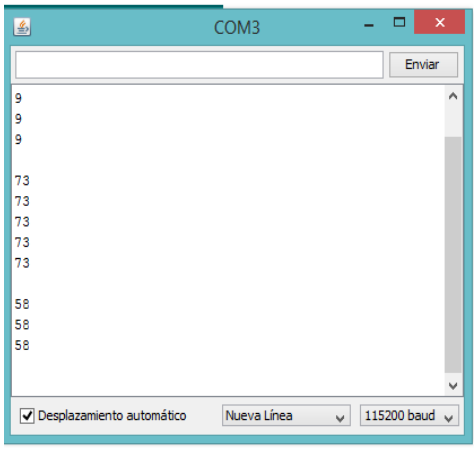


Figura 4.12 Demostración de transmisión de datos

Finaliza entonces la rutina transmite con un cambio de estado de la variable bandera2 de verdadero a falso, para dar paso a la rutina de recibe ();

En el caso de la rutina recibe (); está condicionada de la siguiente manera:

Figura 4.13 Rutina recibe ()

- Si no has recibido una respuesta, entonces sigue con la transmisión del último número tecleado (a menos que el usuario disponga lo contrario y detenga la transmisión).

- Si recibes un “Voy”, activa el pin 13 (encender el buzzer) en 5 ocasiones con intervalos de 500 ms indicando que el colectivo se detendrá, limpia todas las variables y vuelve a condicione iniciales en espera de otro número.

- Si recibes un “No” activa el pin 13 (encender el buzzer) en 10 ocasiones con intervalos de 100 ms indicando que el colectivo no se detendrá, vuelve a condicione iniciales y el usuario debe teclear de nuevo el numero deseado.

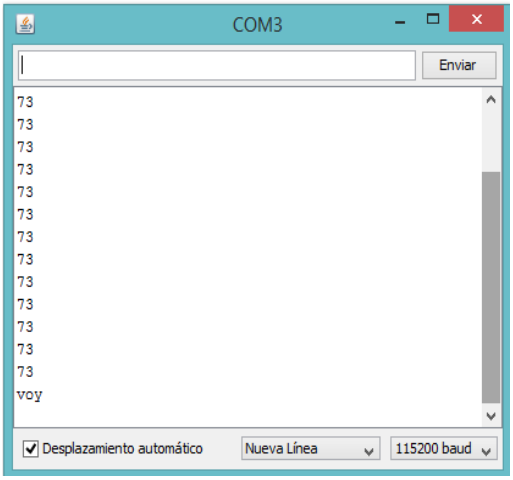


Figura 4.14 Respuesta positiva de la combi

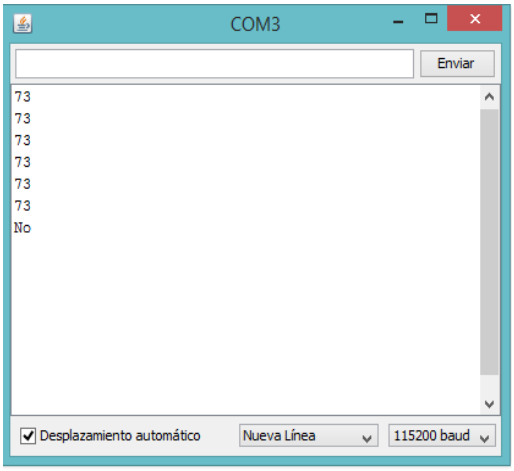


Figura 4.15 Respuesta negativa

Por último tenemos la rutina reset (), la cual genera la interrupción externa cuando el usuario pulsa el botón para resetear el "control", borrando cualquier contenido en las variables y regresando a condiciones iniciales en espera del siguiente numero.

```
void reset()
{
  x=0; texto="";
bandera=false;
bandera2=true;
}
```

Figura 4.16 Rutina de interrupción externa

A continuación se presenta el diagrama de flujo del dispositivo maestro, con las condiciones que pudiesen presentarse cuando el usuario llega a la parada del transporte.

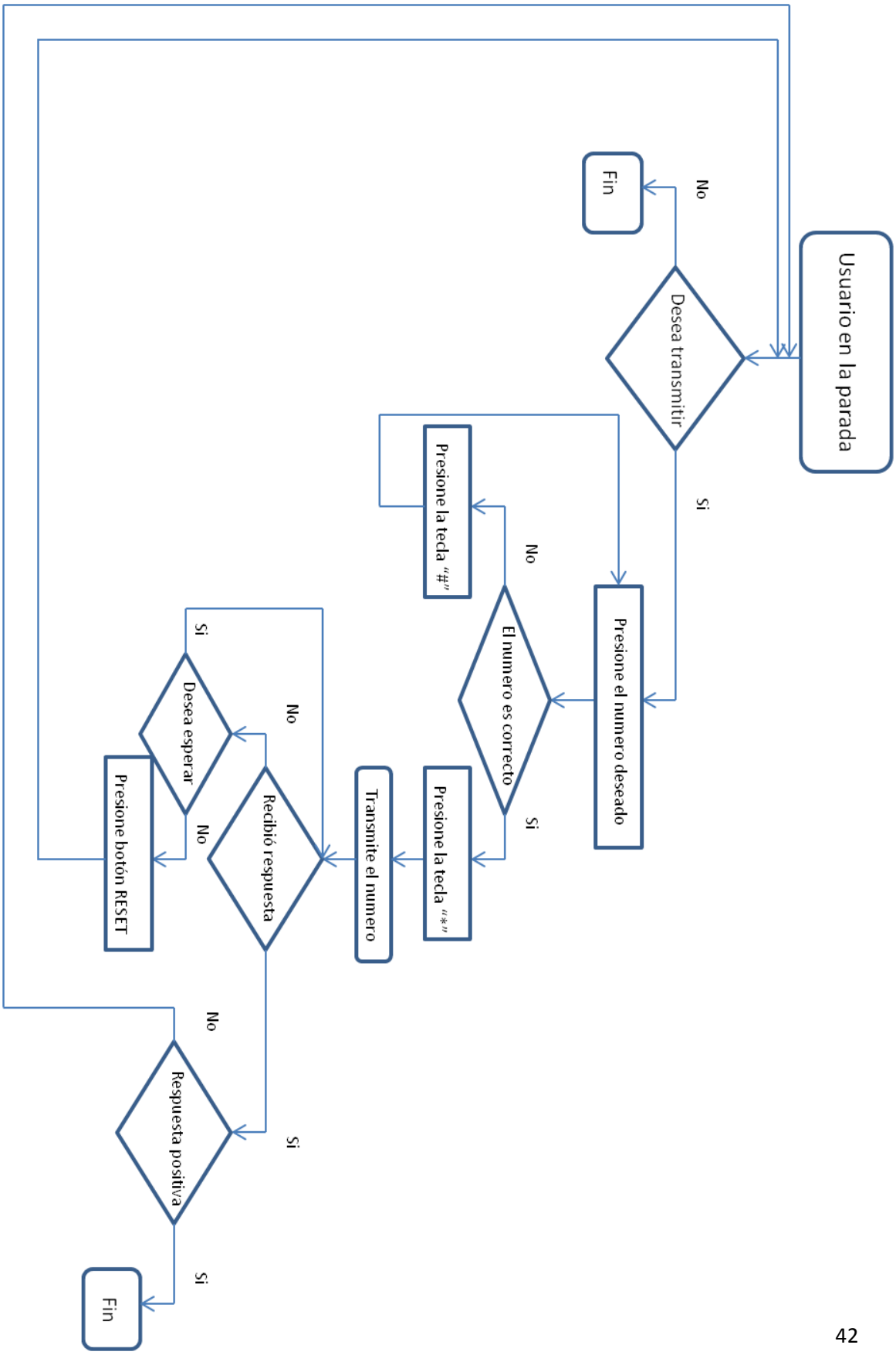


Figura 4.17 Diagrama de Flujo dispositivo maestro

4.6.2 Código para dispositivo esclavo (“Colectivo”)

Después de haber explicado a detalle el código utilizado para la transmisión de datos por parte del usuario hay que mencionar que por parte del colectivo algunas líneas de código coinciden, como es el caso de la configuración del canal de comunicación y las direcciones de transmisión y recepción.

Al igual que el maestro, el esclavo presenta subrutinas para facilitar la estructura del código y reducir un poco las líneas de código, puesto que hacerlo en un mismo loop traería problemas ya sea de confusión del programador y a su vez un ligero retardo en la codificación por parte del microcontrolador pudiendo hacer un poco más lento el proceso de transmisión.

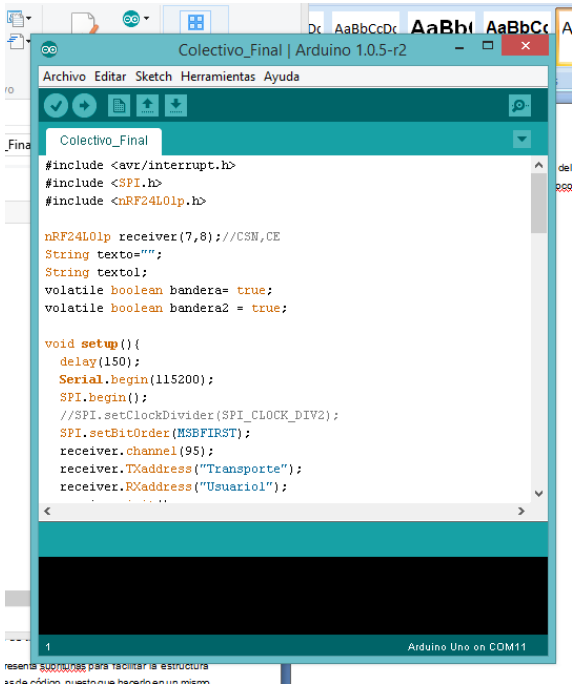


Figura 4.18 Declaración de variables del dispositivo receptor

De igual manera tenemos las librerías necesarias para entablar una comunicación y para generar una interrupción externa. La declaración de variables es la siguiente:

String texto. Al igual que en el otro dispositivo debe declararse puesto que es la variable que vamos a recibir del usuario.

String texto1. En este caso es la variable que vamos a enviar con la respuesta positiva o negativa.

Volatile boolean bandera y boolean bandera 2. Son banderas que permitirán el switcheo de las rutinas, en este caso pueden o no ser volátiles, esto significa que se almacenan en la memoria RAM del microcontrolador y pueden ser borradas y reescritas sin problemas.

Pasamos al setup () y encontramos la mismas líneas puesto que son la configuración del modulo, con la única diferencia que ya no lo maneja como "transmitter" si no como "receiver" puesto que es el receptor del transceptor.

Como nota, el canal debe ser el mismo en ambos lados, así como también las direcciones.

Antes de pasar al loop del código explicaré las subrutinas que conforman el código.

Colectivo_Final

```
void obtener()
{
  if(receiver.available()>0)
  {
    receiver.read();
    receiver.rxPL(texto);
    Serial.println(texto);
    if (texto == "73")
    {
      digitalWrite(6,1);
      texto="";
      bandera=false;
    }
    //texto="";
    }

else{texto="";}
}
```

Figura 4.19 Rutina obtener () la cual recibe el dato del usuario

Con esta rutina como coloquialmente se dice “bajamos” el dato que envía el usuario, la función read () nos permite leer el dato que recibimos el cual está en la variable *texto* y lo comparamos, en caso de ser un 73 procedemos a encender el led ubicado en el pin 6 del arduino UNO y hacemos un cambio de estado del a variable *bandera* de verdadero a falso para pasar a la transmisión, de no ser un 73 la combi no hace nada.

Pasamos ahora a la transmisión.

```
void transmite()
{
receiver.init();
receiver.txPL(textol);
receiver.send(SLOW);
delay(50);
Serial.println(textol);
//textol="";
bandera=true;
bandera2=true;
receiver.init();
}
```

Figura 4.20 Rutina de transmisión

En la rutina transmite (), después de comparar que es un 73 las condiciones son las siguientes:

- Si la respuesta es positiva transmitirá un “voy”, apagará el led y volverá a condiciones iniciales para esperar algún otro dato.
- Si la respuesta es negativa transmitirá un “no” apagará el led y volverá a condiciones iniciales en espera de otro dato.
- Cumplida cualquiera de las dos condiciones anteriores limpiará el buffer del modulo para no tener algún dato “basura”.

Los dos game bottoms que tiene el colectivo generan cada uno una interrupción como a continuación se presenta.

```
void responder()
{
  if(digitalRead(6) == 1)
  {
  digitalWrite(6,0);
  textol="voy";
  bandera2=false;
  }

}
```

Figura 4.21 Interrupción para respuesta positiva

```
void rutina2()
{
  if(digitalRead(6)==1)
  {
  digitalWrite(6,0);
  textol="No";
  bandera2=false;
}
}
```

Figura 4.22 Interrupción para respuesta negativa

A continuación se presenta el diagrama de flujo del dispositivo esclavo que va al colectivo.

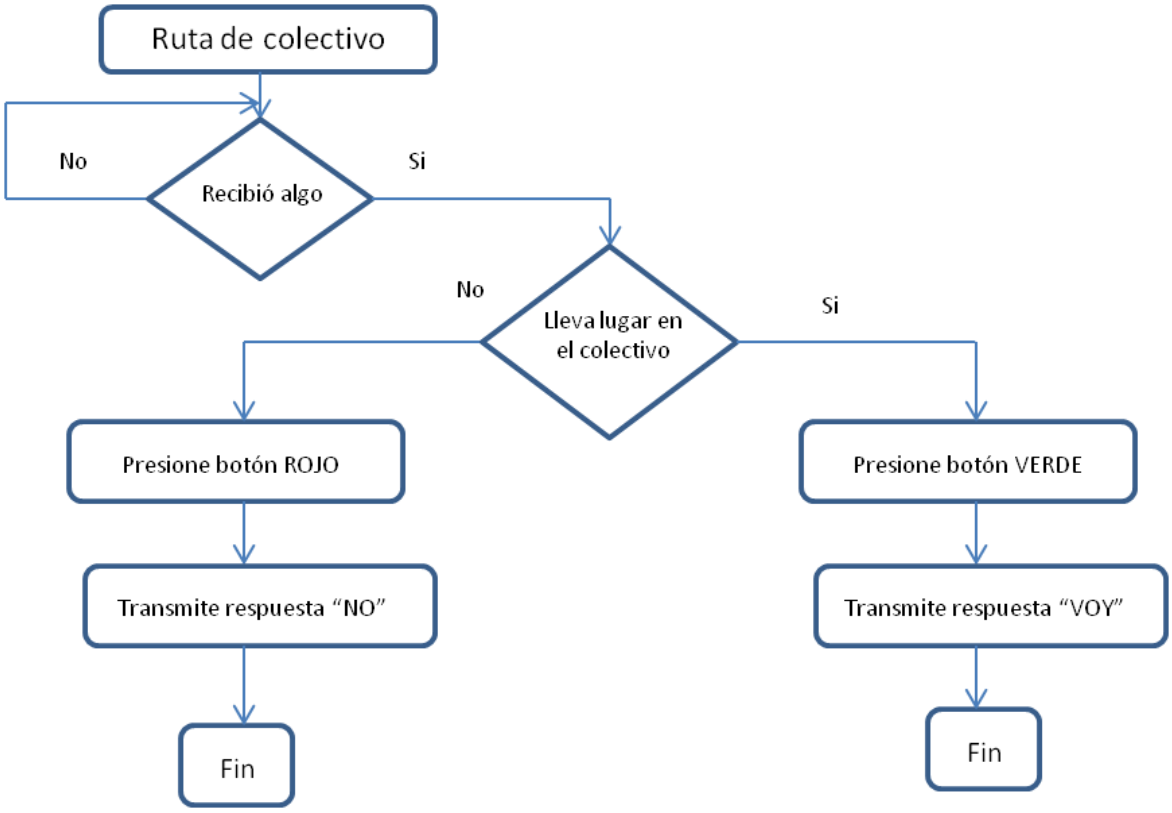


Figura 4.23 Diagrama de flujo dispositivo Esclavo

RESULTADOS

En cuanto a la comunicación mediante el modulo nRF24L01, podemos decir que el resultado es favorable, entrega en línea de vista directa una distancia de entre 80 y 90 metros, el datasheet marca que son 100 mts, técnicamente estamos cerca de la distancia ideal del modulo. Por otra parte, al hacer las pruebas en el medio al que se aplicara el SIT, la atenuación de la señal llega al 50%, dejándonos un rango de entre 30 y 40 mts, en los que la recepción es medianamente buena, sin obstruir con el trabajo efectivo del sistema.

De cada 10 iteraciones realizadas a diferentes distancias, siempre y cuando esta no sobrepasara el margen de los 30-40 metros las 10 iteraciones eran recibidas y respondidas por la otra parte. Una desventaja que podemos percibir es el consumo de corriente del modulo, que a pesar de ser de ultra bajo consumo como lo mencione en la parte teórica, en la practica el modulo consume por cada transmisión, por cada recepción e incluso en stand-by lo que genera que el desgaste de la batería sea considerable por ello se agrego un interruptor el cual corta el flujo de energía en todo el dispositivo, así para ahorrar batería (la cual de igual forma es recargable) el usuario podrá apagar el sistema y encenderlo cuando desee utilizarlo de nuevo.

Otro inconveniente del sistema es el tipo de radiación de la señal, puesto que esta emite en forma de radio abarca la misma distancia 360 grados alrededor del usuario, lo cual traería consecuencias en calles muy angostas. Se probó aislar una parte del contendor del modulo, ya que algunos blogs mencionan que el aluminio es una buena forma de interferir con la recepción de señales, sin embargo no funciono de la mejor manera.

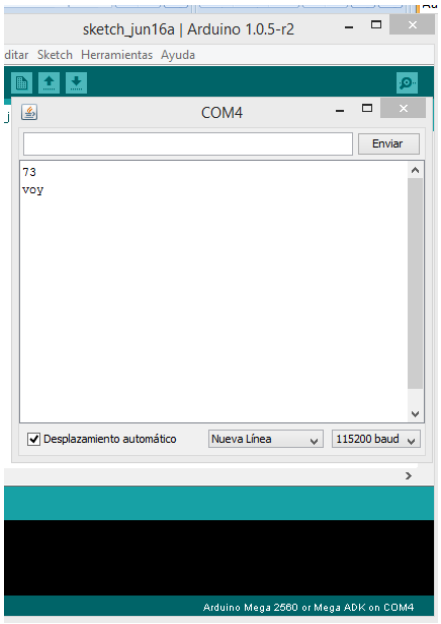
Fuera de eso el sistema corre a la perfección e incluso podría ser usada esta misma base con otros fines posteriores, solo es cuestión de encontrar la idea y plasmarla en algo tangible como el Sistema de Identificación de Transporte, que sin duda parece ser una idea prometedora para las personas con discapacidad visual, siendo ellos los que evaluaron el prototipo y dieron su visto bueno, y es

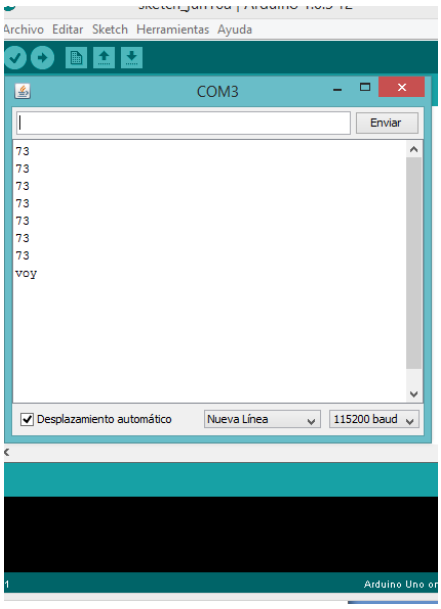
gracias a ellos que se pudo aplicar a las teclas una mica especial con la cual se escribe en braille.

Con ayuda de una persona con discapacidad visual se grabo en esta mica el valor correspondiente a cada tecla, el lenguaje braille es un poco complicado para personas que no tenemos la practica suficiente para entenderlo, sin embargo nos explicaron que los números llevan un símbolo, al que ellos denominan “símbolo numérico” con el cual identifican que los puntos marcados son un numero y no una letra, puesto que en el lenguaje braille las primeras 9 letras corresponden a los números del 1-9 y este símbolo les permite distinguirlo.

Un par de personas probaron el sistema y están de acuerdo en usarlo siempre y cuando se perfeccione y se ponga en marcha.

A continuación se presentan unas imágenes de lo que se ve en el monitor serial de Arduino en ambas partes del sistema tanto en el colectivo como en el usuario.





Figuras 5.1 y 5.2 Monitor serial Colectivo (izq.), Monitor serial usuario “Control” (der.)

Hay que recalcar, que en este prototipo se especifico la ruta 73, por poner un ejemplo, así puede ser la ruta 1 o la 2 o cualquiera que uno desee, incluso todas las de la ciudad, siempre y cuando cuenten con el dispositivo que va en el colectivo, el sistema es versátil, innovador y sobretodo adaptable a cualquier ciudad. Además de que no es únicamente para personas con discapacidad visual, sino para débiles visuales, ciegos parciales, personas de la tercera edad, personas con miopía o que simplemente no tengan una percepción visual correcta a una distancia lejana.

CONCLUSIONES Y RECOMENDACIONES

Después del desarrollo del proyecto que se complico debido a que el modulo no es tan comercial como pareciera y puesto que todos los ejemplos y librerías están en otros idiomas, podemos concluir que es posible realizar este proyecto, en verdad hacerlo de manera más formal y considerar puntos que quizá a nosotros nos faltaron, la tecnología está ahí, solo es cuestión de tomarla y decidirse por hacer algo con ella.

Al cambiar la librería de Mirf.h a nRF24L01p.h optimizamos quizá una semana de tiempo puesto que los comandos eran más entendibles y fáciles de manejar.

Arduino es una plataforma que te simplifica muchas cosas, basta conectar tu cable USB par empezar a programar, los resultados obtenidos eran los esperados ya que se sabía que al llevar al medio exterior un modulo que trabaja mediante RF, y a 2.4GHz es evidente que la atenuación de la señal será un problema, pero un estándar de 30 metros era lo que buscábamos y lo conseguimos, quizá hizo falta un poco de presupuesto como para probar la versión de larga distancia de este mismo modulo, la cual ya integra una antena para extender su capacidad de transmisión.

Actualmente trabajamos con 2dBm para la transmisión aunque el modulo maneja también 0dBm lo cual nos daría un resultado menos favorable al reducir la potencia de transmisión.

Como una recomendación importante y esencial, quizá la única que podríamos dar es que el modulo es bueno, es eficaz y es efectivo, pero sería bueno poder eliminar la antena que tiene incluida y diseñar una de tipo direccional, para evitar el problema de interferencia con la acera de enfrente, por falta de tiempo y porque no pudimos cubrir ese detalle no se corrigió ese problema pero considero que es mínimo con respecto a los demás que se nos presentaron.

La programación queda abierta a modificaciones y lo que es mejor es que queda libre a cualquier ruta, el usuario puede teclear la ruta que desee, desde la 1 hasta

la 999 en caso de que existiera, y así como está limitada a 3 cifras puede hacerse a 4 o 5 o más.

En conclusión, considero que los objetivos planteados se lograron y los problemas por resolver se resolvieron, no puedo afirmar que un invidente pueda confiar su vida a este sistema puesto que le faltan ciertas delimitaciones al protocolo y la cuestión de la forma de radiación del modulo, pero es un paso importante de una idea que surgió de la nada y ahora está siendo materializada.

COMPETENCIAS APLICADAS

A lo largo de los 9 semestres de la carrera de electrónica pudimos ir conociendo las diferentes competencias aplicables en la carrera, como son el aprendizaje individual y en equipo, la interdisciplinariedad de la carrera con otras áreas de la ingeniería.

Es por ello que de la materia de Microcontroladores, las competencias aplicadas fueron las siguientes:

- Conocer los fundamentos básicos de una comunicación serial síncrona y asíncrona.
- Comunicación SPI.
- Puertos de Entrada/Salida digital.
- Interrupciones.
- Programacion de puertos digitales.
- Programacion en lenguaje ensamblador.
    - Conjunto de Instrucciones.
    - Instrucciones de control de programa
- Estructura del Programa.

En cuanto a la materia de programación estructurada, las competencias aplicadas fueron:

- Utilizar la metodología de programación modular para desarrollar programas estructurados y simplificar el mantenimiento del código.
- Desarrollar programas que incluyan manejo de puertos para permitir la interacción con sistemas electrónicos externos a la computadora.
- Utilizar correctamente las herramientas de diseño para el desarrollo de programas.

- Desarrollar programas a partir del diseño de algoritmos para su aplicación en la solución de problemas.

En cuanto a comunicaciones se utilizaron las competencias marcadas en la asignatura de tópicos selectos de comunicación:

- Conocer las bandas de comunicaciones actuales y futuras.
- Conocer la política y reglamentación de las telecomunicaciones.

Con respecto a la materia de Taller de investigación I y II las competencias son:

- Buscar y clasificar los diferentes tipos de investigación en el ámbito científico y tecnológico fuera de la institución.
- Elaborar un protocolo de investigación en el área de su formación profesional, de acuerdo a los lineamientos establecidos.

BIBLIOGRAFIA

- http://robotica.wordpress.com/about/
- https://sites.google.com/site/proyectosroboticos/nrf24l01
- http://www.techmake.com/wrl-00030.html
- http://www.electrodragon.com/w/index.php?title=NRF24l01
- http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P
- http://www.slideshare.net/hugoangaritaespinosa/libro-kit-basico.

- *Arduino Curso Práctico de Formación*
  Oscar Torrente Artero
  Editorial RC libros 2013

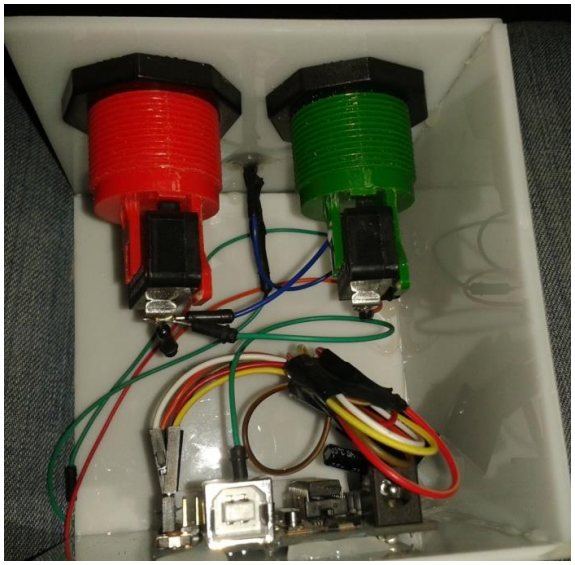
- *30 Arduino projects for the Evil Genius*
  Simon Monk
  Editorial Mc Graw Hill 2010

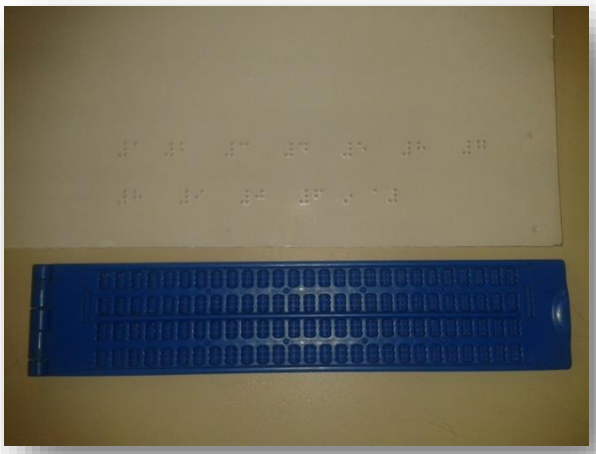
- *Arduino Cookbook*
  Michael Margolis
  Editorial O´reilly 2011

- *Arduino Programming notebook*
  Brian W. Evans
  Sin Editorial Edición Septiembre 2008

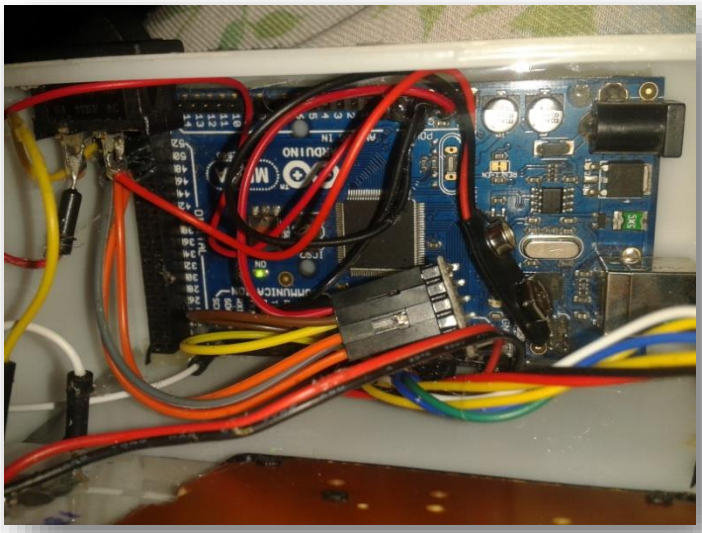
ANEXOS

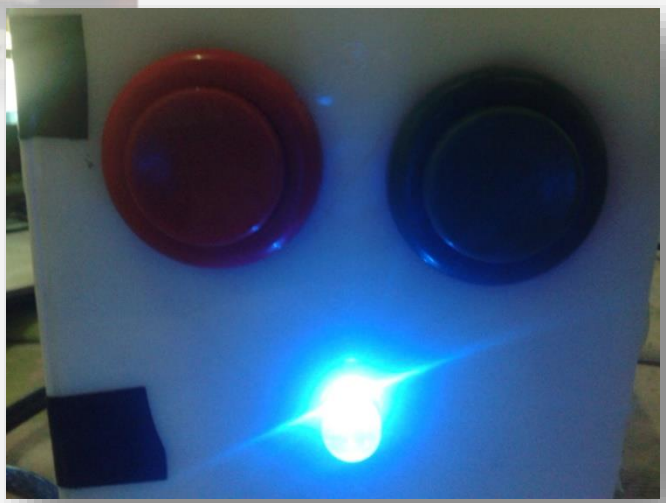
Imágenes de ambos dispositivos





Tablilla para escribir en braille





Anexo Codigo en arduino para dispositivo maestro (Control)

```
#include <SPI.h>
#include <nRF24L01p.h>
#include <Keypad.h>
#include <avr/interrupt.h>

nRF24L01p transmitter(7,8);//CSN,CE
char key;
String texto="";
String texto1="";
int x=0;
boolean bandera=false;
boolean bandera2=true;
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {10,9,6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
```

```
  delay(150);
  Serial.begin(115200);
  SPI.begin();
  //SPI.setClockDivider(SPI_CLOCK_DIV2);
  SPI.setBitOrder(MSBFIRST);
  transmitter.channel(95);
  transmitter.RXaddress("Prado");
  transmitter.TXaddress("Artur");
  transmitter.init();
  pinMode(20, INPUT);
  digitalWrite(20, HIGH);
  attachInterrupt(3,reset,FALLING);
}

void loop()
{
  if(bandera==false )
  {  get(); }

  if (bandera==true && bandera2==true)
  {
    transmite();
    delay(50);
    //transmitter.init();
    //bandera2=false;
  }
```

```
  if(bandera==true && bandera2==false)
   {
    recibe();
    transmitter.init();
    delay(50);
}}

void get()
  {  do{
      key= keypad.getKey();}
      while(key==NO_KEY);
     x=x+1;
       switch (x)
         {
           case 1:
           case 2:
           case 3:
           case 4:

   if(key=='*')
    {
      bandera=true;
    }
   else
    {
      if(key=='#')
        {x=0;texto= "";}
```

```
   else texto = texto + key;
     }
   break;

   default: x=0; texto="";
  }
    delay(20);
}

void transmite()
  {
    transmitter.txPL(texto);
    transmitter.send(SLOW);
    delay(50);
    Serial.println(texto);
    bandera2=false;
}

void recibe()
  {
    bandera=true;
    bandera2=true;
       if(transmitter.available())
     {
        transmitter.read();
        delay(300);
        transmitter.rxPL(texto1);
```

```
    Serial.println(texto1);
if(texto1 == "voy")
 {
   for(int i=1; i<=5; i++)
   {
  tone(13,2000);
 delay(500); noTone(13); delay(500);
   }
noTone(13);

   texto=""; x=0; texto1="";
    bandera = false;
    bandera2=true;
 }

 if(texto1=="No")
 {
   texto=""; texto1=""; x=0;
   for(int i=1; i<=10; i++)
   {
  tone(13,2000);
 delay(100); noTone(13); delay(100);
   }
noTone(13);

 bandera = false;
 bandera2= true;
```

```
   } } }

void reset()
{
  x=0; texto="";
bandera=false;
bandera2=true;
}
```

Anexo Código para dispositivo esclavo (Colectivo)

```
#include <avr/interrupt.h>
#include <SPI.h>
#include <nRF24L01p.h>

nRF24L01p receiver(7,8);//CSN,CE
String texto="";
String texto1;
volatile boolean bandera= true;
volatile boolean bandera2 = true;

void setup(){
  delay(150);
  Serial.begin(115200);
  SPI.begin();
  //SPI.setClockDivider(SPI_CLOCK_DIV2);
  SPI.setBitOrder(MSBFIRST);
```

```
  receiver.channel(95);
  receiver.TXaddress("Transporte");
  receiver.RXaddress("Usuario1");
  receiver.init();
  pinMode(2, INPUT);
  digitalWrite(2, HIGH);
  pinMode(3, INPUT);
  digitalWrite(3, HIGH);
  pinMode(6,OUTPUT);
  attachInterrupt(0,responder,FALLING);
  attachInterrupt(1,rutina2,FALLING);
}
void loop()
{
  if(bandera==true)
  {
    obtener();
    receiver.init();
  }

  if(bandera==false && bandera2==false)
{
  transmite();
  receiver.init();
}}

void responder()
```

```
{
  if(digitalRead(6) == 1)
  {
  digitalWrite(6,0);
  texto1="voy";
  bandera2=false;
  }}

void transmite()
{
receiver.init();
receiver.txPL(texto1);
receiver.send(SLOW);
delay(50);
Serial.println(texto1);
bandera=true;
bandera2=true;
receiver.init();
}

void obtener()
{
  if(receiver.available()>0)
  {
    receiver.read();
    receiver.rxPL(texto);
    Serial.println(texto);
```

```
    if (texto == "73")
    {
      digitalWrite(6,1);
      texto="";
      bandera=false;
    }   }
else  { texto=""; }
}

void rutina2()
{
  if(digitalRead(6)==1)
  {
  digitalWrite(6,0);
  texto1="No";
  bandera2=false;
}}
```