Informe
Practicas
Profesionalizantes

Fabian Rincon Martinez

Noviembre 2020

Índice

Nombres y Apellidos: Martinez Rincon Fabian, Pablo Poblete, Pergamo Leonel

Nombre del trabajo: Botonera Arduino

Grupo de Trabajo: Sector Arduino

Supervisores: Gastón Valdez y Camilo Di Paolo

Modalidad de trabajo: Práctica Formativa Obligatoria

Hora de encuentros: viernes de 15:00 a 16:30

Periodo del proyecto: 1/4/2020 hasta 16/11/2020

Introducción

En este informe se mostrará y describirá todo lo cumplido por el alumno Fabian Martinez Rincon de la Escuela Técnica numero 6 Albert Thomas, en el sector Arduino. Durante el periodo de prácticas aprenderemos el uso y funciones de las herramientas y módulos que nos brinda Arduino y como aprovecharlas a lo largo de nuestros proyectos. Vamos a realizar una botonera en el cual mediante dos teclas nos permite subir y bajar en un abecedario y poder elegir letra por letra hasta enviar una frase por bluetooh a nuestro receptor que se encuentra en el lugar deseado por el usuario.

Componentes

Componentes	Cantidad
Arduino UNO	2
LCD OLED i2C 128x64	1
LCD 16X2 SPI	1
Pulsadores	4
Modulo Bluetooth HC-06	1
Modulo Bluetooth HC-05	1
Cargador de celular (5V)	2
Buzzer	1

Cuadro 1:1 Tabla de componentes

Arduino

Figura 2: Arduino físico

Descripción

Arduino es una plataforma de hardware libre basada en una placa de entradas y salidas en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring. Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador. Las placas como la que se muestra en la figura 2 se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

La mayoría de las placas Arduino constan de un microcontrolador AVR Atmel-8 bits (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560), cada microcontrolador consta de diversas cantidades de memoria flash, pines y funciones. Las placas utilizan pines/cabezales hembra de una o dos hileras que facilitan las conexiones e incorporación en otros circuitos.

Las placas Arduino pueden conectarse con módulos adicionales denominados shields (escudos, por su traducción al español), dichos shields aumentan las características técnicas de la placa Arduino en uso, debido a que poseen circuitos específicos que añaden una o más funcionalidades extras a la placa Arduino nativa en la cual se utilice, también se les conoce como placas de expansión.

Modulo Bluetooth HC-06/HC-05

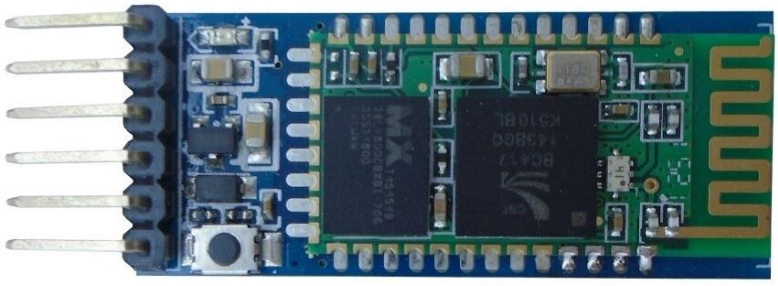


Figura 2:2

Módulos Bluetooh

Descripción

El módulo HC-05 que se muestra en la figura 2:2 ofrece el servicio de puerto serie (RFCOMM), creando un enlace de datos transparente entre una PC,

celular, Tablet o cualquier dispositivo con Bluetooth y el microcontrolador. La salida del módulo es una señal serial asíncrona que puede ser recibida e interpretada fácilmente por cualquier microcontrolador.

Memoria EEPROM del Arduino UNO <EEPROM.h>

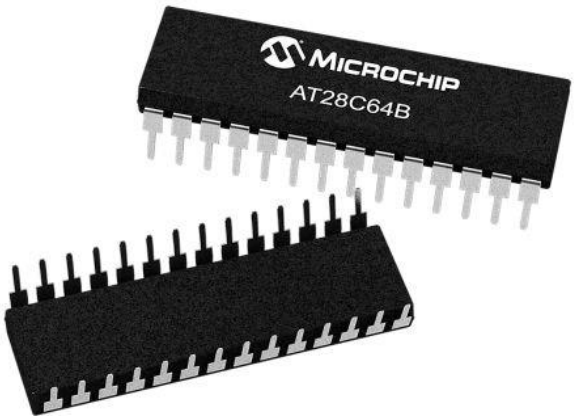


Figura 3: Memoria EEPROM

En la figura 3 se muestra la memoria EEPROM (del inglés Electrically Erasable Programmable Read-Only Memory = ROM programable y borrable eléctricamente) es una memoria no volátil. Esto significa que los datos que almacena no se pierden al desaparecer la alimentación de un dispositivo.

La EEPROM fue pensada para mantener aquellos datos que deseamos resguardar luego de apagar y reiniciar un microcontrolador, y disponer de ellos al retomar la operación.

El ATmega328P —el microcontrolador del Arduino UNO, el Nano y otros de la línea Arduino— tiene una capacidad de 1.024 bytes (1 Kb) de memoria EEPROM. La hoja de datos nos indica que la EEPROM del ATmega328P acepta hasta 100.000 ciclos de lectura/escritura. Parece mucho, pero el solo hecho de tener una cantidad de ciclos de escritura acotada ya nos indica que su función no es en nada similar a una RAM. Es para guardar datos, y no en forma momentánea y veloz, sino de manera durable. No ponga nunca operaciones con la EEPROM dentro de la función loop (), ya que, dada la velocidad de trabajo del microcontrolador, llegará rápidamente al límite de 100.000.

En general, los distintos microcontroladores en las distintas tarjetas de Arduino contienen diferentes cantidades de EEPROM:

– 4kb (4096 bytes) en ATmega1280 y ATmega2560.

– 1kb (1024 bytes) en ATmega328P.

– 512 bytes en Atmega168 y ATmega8.

– 1kb (1024 bytes) emulados en Arduino 101

Pantalla OLED 128X64 con I2C (ssd1306) <Adafruit_SSD1306.h> <Adafruit_GFX.h>

Descripción General

Esta pantalla es muy pequeña (1.3 pulgadas de diagonal) pero muy visible dado su alto contraste OLED. Es una pantalla con una matriz de un color de 128x64 puntos. Dado que la pantalla está basada en la tecnología LED, no necesita retroiluminación y tiene un alto contraste incluso a plena luz del día, En la figura 4 se muestra una imagen de su funcionamiento.

El driver interno es un SH1106 que se comunica por I2C, un protocolo muy rápido ideal para este tipo de pantallas. Internamente todo el conjunto funciona a 3,3V pero se han acoplado tanto la alimentación como los pines de entrada para funcionar perfectamente a 5V lo que lo hace ideal para utilizar con nuestro microcontrolador favorito de 5V!

Características:

Tamaño: 0.96 pulgadas.

Resolución: 128X64

Color: azul y amarillo.

Soporte amplio voltaje: 3.3V-5V DC.

Temperatura de trabajo: -30-80 grados.

Interfaz

Figura 4: Interfaz de la pantalla Oled

Pulsadores:

El dispositivo principal cuenta con 4 pulsadores como los que se muestran en la figura 5 (Subir, Bajar, Confirmar Letra y confirmar opción/palabra). La idea es utilizar la menor cantidad de pulsadores para que sea compacto y sencillo de usar.

Partiendo de la definición de interruptor como un dispositivo que cuando se encuentra “cerrado” deja pasar la corriente y cuando está “abierto” impide el paso de la misma, un pulsador no es más que un tipo de interruptor, el cual se mantendrá en posición de “cerrado” tanto tiempo mientras lo mantengamos pulsado, gracias a la lámina conductora que produce el contacto.

El Arduino UNO cuenta con resistencias internas de 20kΩ, por ende, las vamos a aprovechar por medio de la línea de código (INPUT_PULLUP)

Figura 5: Pulsadores

Buzzer

En la figura 6, se muestra un pequeño transductor capaz de convertir la energía eléctrica en sonido. Para hacerlos funcionar basta con conectar el positivo con el + y la tierra o negativo con el – de una batería o cualquier fuente de corriente directa. está ubicado en el Arduino receptor y sirve para alertar cuando enviamos un dato desde el Arduino principal.

Figura 6: Buzzer

Diagrama de flujo general

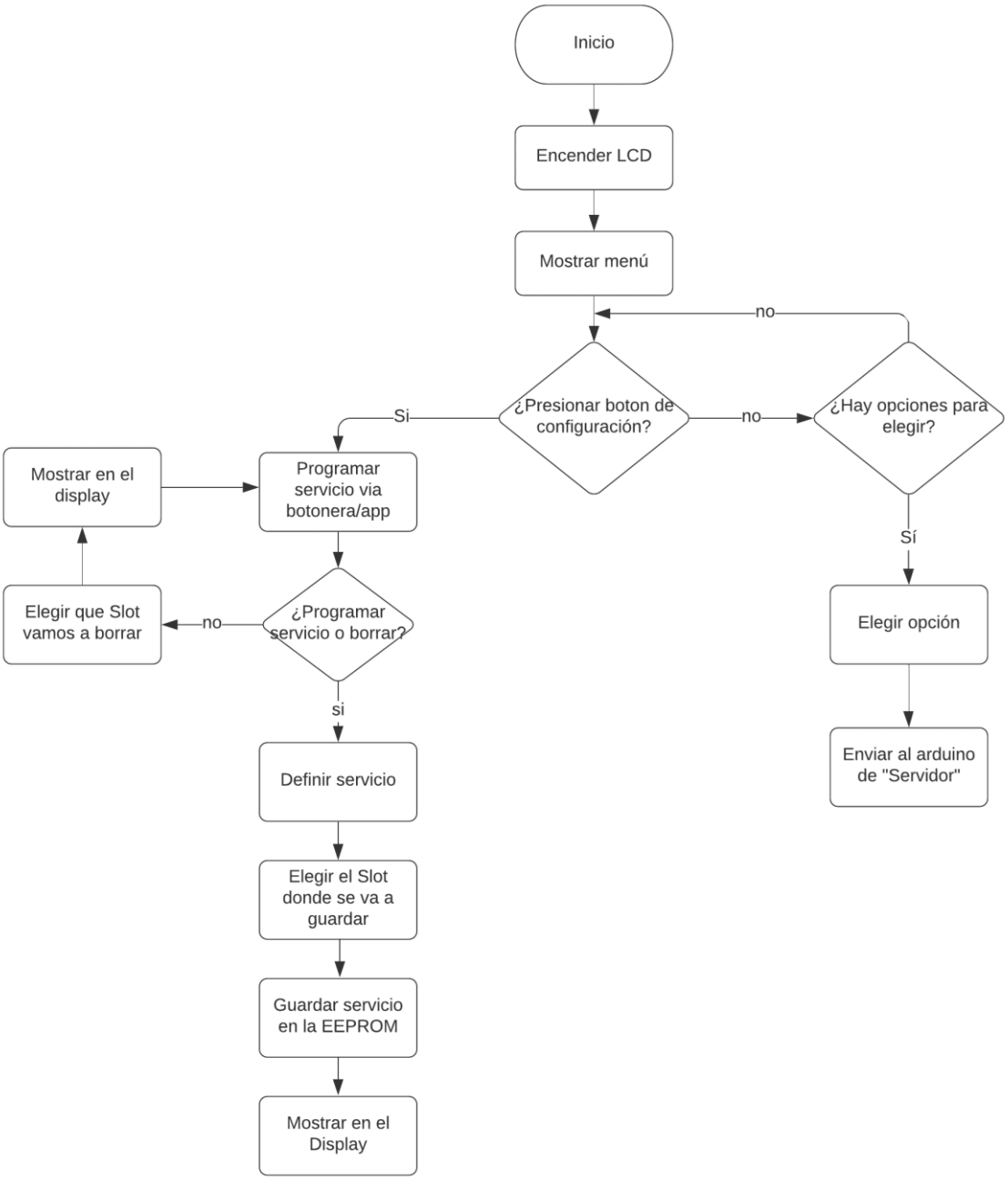


Figura 7: Diagrama de Flujo del circuito completo

Diagrama de la Botonera

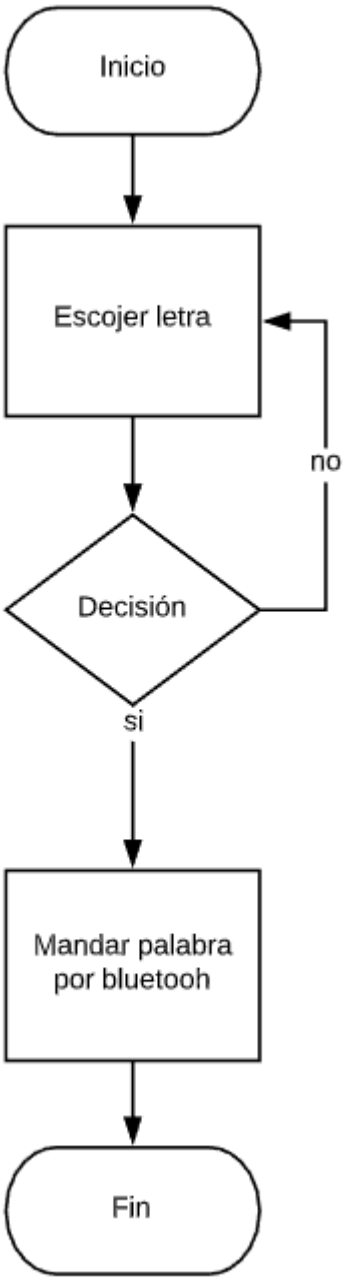


Figura 8: Diagrama por parte de la botonera

Desarrollo.

Al principio programamos los pulsadores para que mande una palabra escrita por el usuario al puerto serial. Para esto el usuario tiene que navegar por el array abc a través de los pulsadores letra por letra hasta encontrar la que el desea y cuando termine poder enviarla. El maestro que será el encargado de mandar por puerto serial estos datos, para que, al momento de ingresar el botón de “terminar palabra”, solo se mande la palabra cargada. Tal y como se muestra en la “figura 10”

Circuito con pulsadores y pantalla serial.

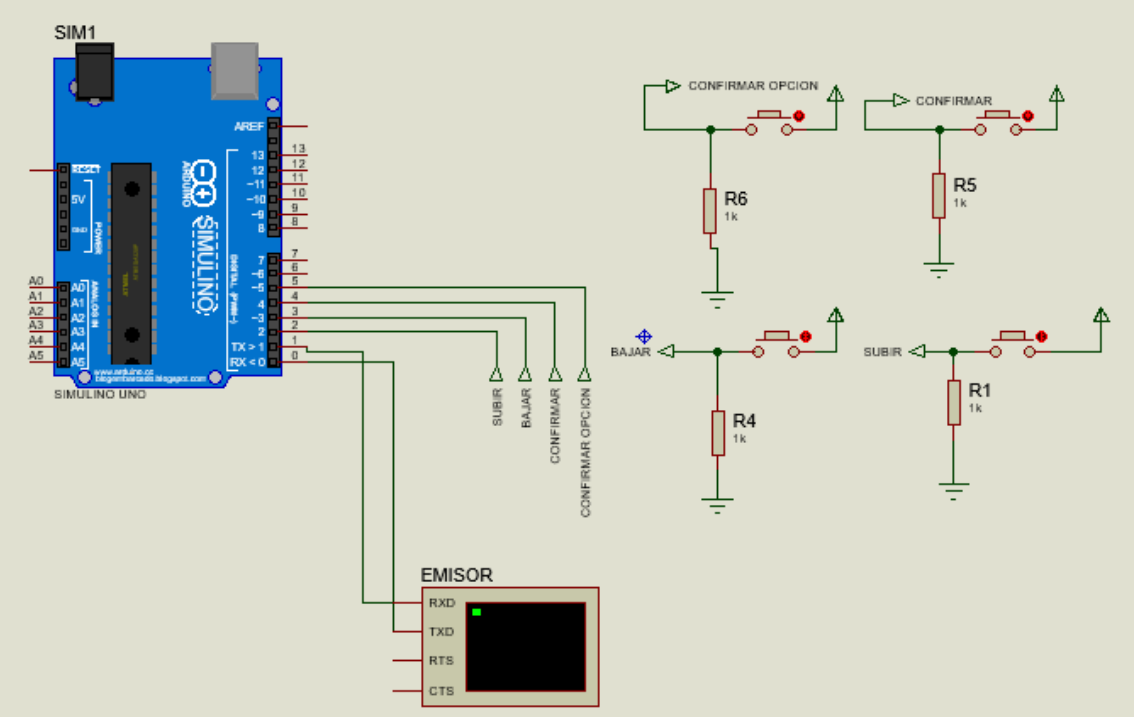


Figura 9: Esquema de control mediante pulsadores pull down

Una vez tenemos el circuito que se muestra en la “figura 9”, para comprobar que funciona, ingresamos cierta cantidad de letras para estar al tanto del funcionamiento (Como se muestra en la figura 10). La letra “ñ” no está en nuestro lenguaje, pero esto lo corregimos más adelante.

En este caso utilizamos el serial para comprobar de que no tenemos errores en el código ya que puede pasar que por errores de conexión tardemos mucho mas en encontrar el error que si solo utilizamos el puerto serial. A demás, mas adelante nos ayudara ya que sabemos que tiene que mostrar por el display y que no tiene que mostrar en caso de errores.

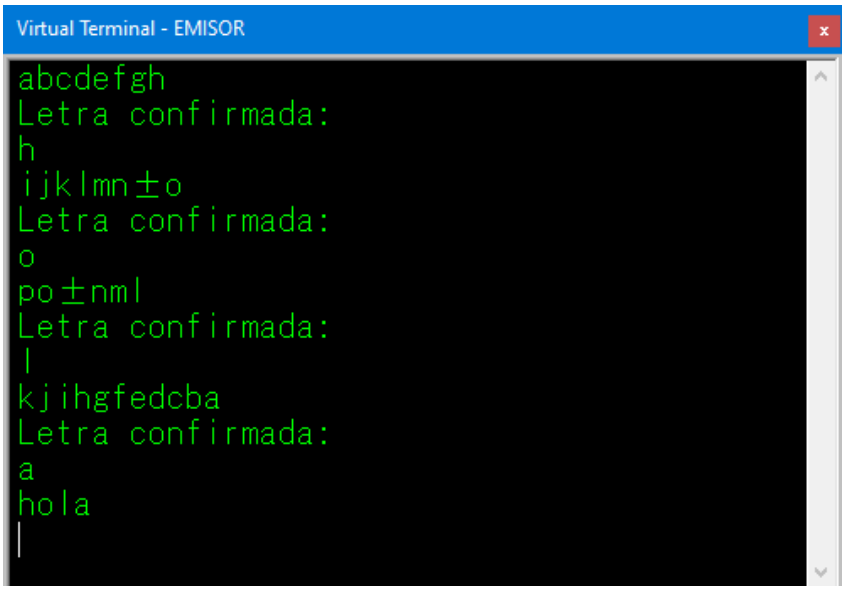


Figura 10: Pulsadores en Funcionamiento.

En la “figura 11” se cambian los pulsadores pull down que estábamos utilizando por pulsadores pull up, ya que nos permitía utilizar las resistencias internas del Arduino y ahorrar en componentes (que en este caso no son necesarias) y el circuito queda mas compacto

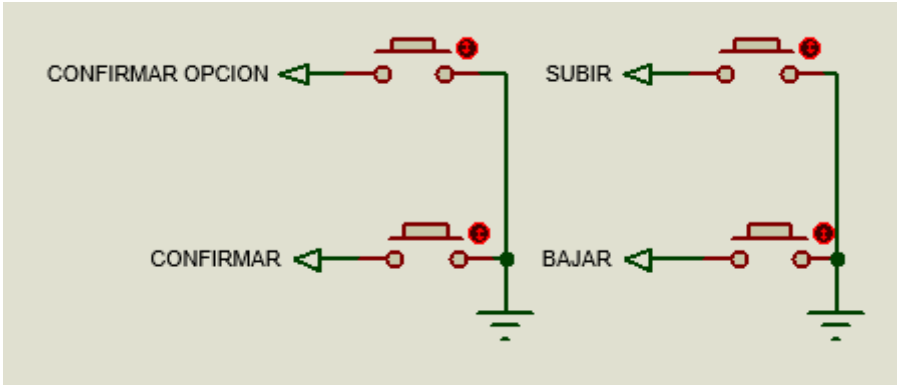


Figura 11: Esquema de Pulsadores Pull Up

Para poder utilizar los pulsadores que tenemos en la “figura 11” necesitamos cambiar la configurar de los pines actuales, ya que estaban en configuración pull down. Para cambiarlos tenemos que configurar los modos en los que Arduino lee los pines (como se muestra en la figura 12), tanto de entrada como de salida. Para que nosotros lo podamos utilizar, tenemos que tener los pines en el modo IPUT_PULLUP ya que nos permite tener 5v siempre a la salida.

```
Serial.begin(9600);
pinMode(SUBIR, INPUT_PULLUP);
pinMode(BAJAR, INPUT_PULLUP);
pinMode(ConfirmarLetra, INPUT_PULLUP);
pinMode(ConfirmarOpcion, INPUT_PULLUP);
```

Figura 12: Programa Interno de los pulsadores

Agregamos los Modulo Bluetooth HC-06 y HC-05 (Como se muestra en la figura 5). En el caso de que solamente tengamos estos módulos en simulación tenemos que crear dos puertos virtuales para que proteus nos permita leer ambos módulos. Utilizamos el programa que se muestra en la “figura 13” para simular estos pines. (Las librerías que utilizamos para los módulos, las dejamos al final del informe)

Figura 13: Simulador de pines

Una vez que terminamos de armar el código y de poder configurar todos los pines necesarios para poder utilizar los módulos bluetooh en proteus, acomodamos las conexiones tanto de los pulsadores, como de los módulos para poder utilizarlo desde ambas pantallas serial. En la “figura 14” tenemos tanto el receptor como el emisor. Y en la “figura 15” tenemos una imagen que nos muestra como es el funcionamiento en ambas pantallas serial.

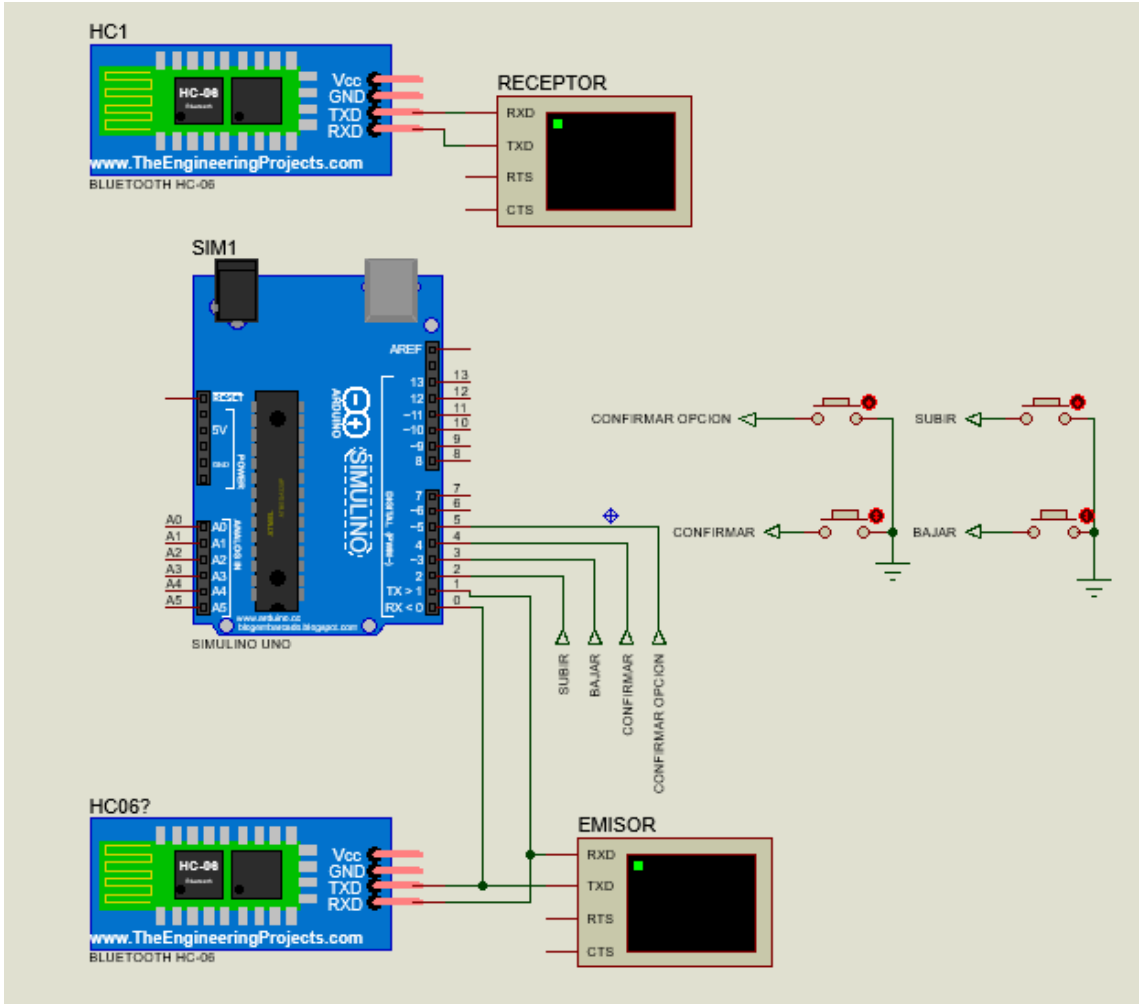


Figura 14: Circuito con pulsadores implementados

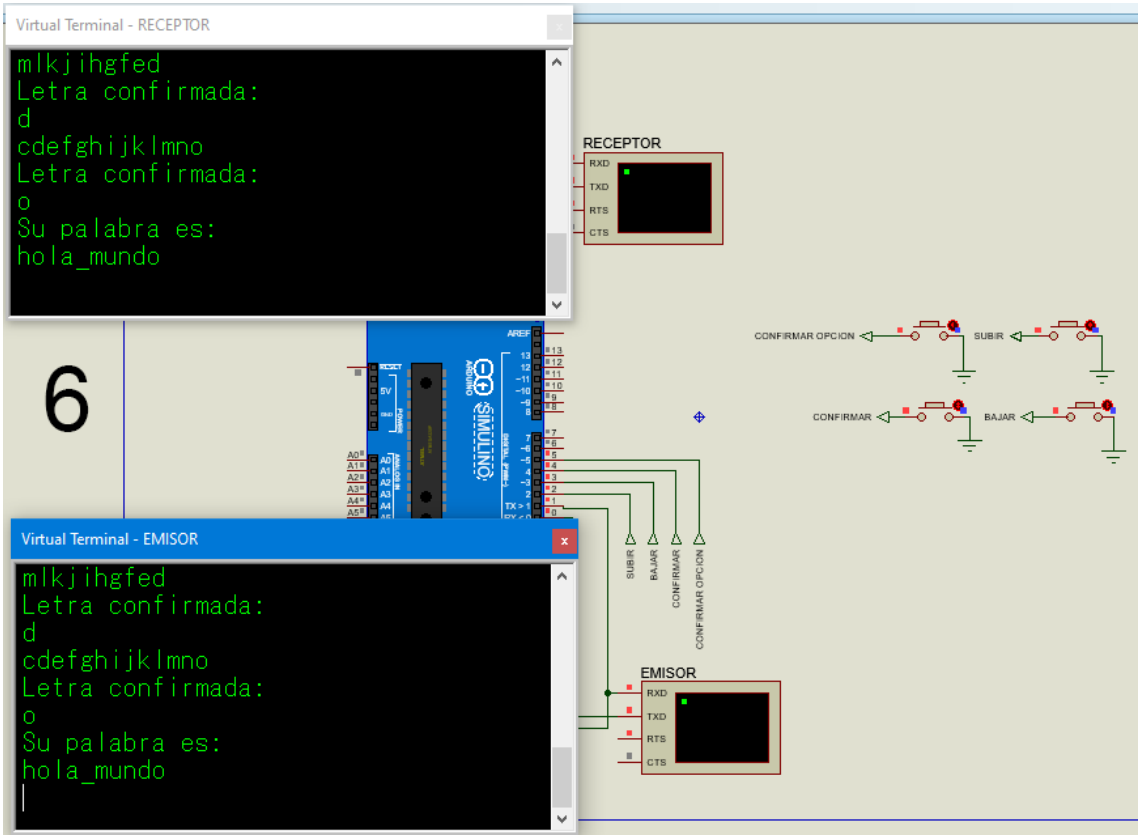


Figura 15: Circuito con puertos serial en simulación

En la “figura 15” imagen tenemos al circuito funcionando y mostrando como se vería en el seria al enviarse una palabra completa. Utilizamos el guion como símbolo de “espacio”. Una vez que se envía la palabra se reinicia.

En la “figura 16” se muestra el que código del pulsador “BAJAR”, en lugar de usar un “delay ()” que es una función bloqueante, utilizamos un contador, el numero en contadorB es 10000 ya que el pulsador iba mas lento que el recorrido del mismo y volvía a ejecutar otra letra. Es por eso que ese es el valor mínimo.

```
void bajar()    //Bajo en el arreglo abc
{

        if((digitalRead(BAJAR)==LOW))
        {
            contadorB++;
            if(contadorB==1)
            {
                letra=tecladoABC[tecla];
                Serial.print(letra);
                tecla--;
            }
            if(contadorB==10000)
            {
                contadorB=0;
            }
        }

  }
```

Figura 16:

Configuración de Pulsadores

Arduino maestro

Por parte del circuito del Arduino maestro, en la “figura 16“se muestra como quedo luego de ingresar el oled que utilizaremos como interfaz y menú para el cliente. En el lcd que se muestra en la “figura 16” se muestra la interfaz y a medida que nosotros pulsamos los interruptores, podemos movernos entre las opciones del menú que hemos cargado con anterioridad.

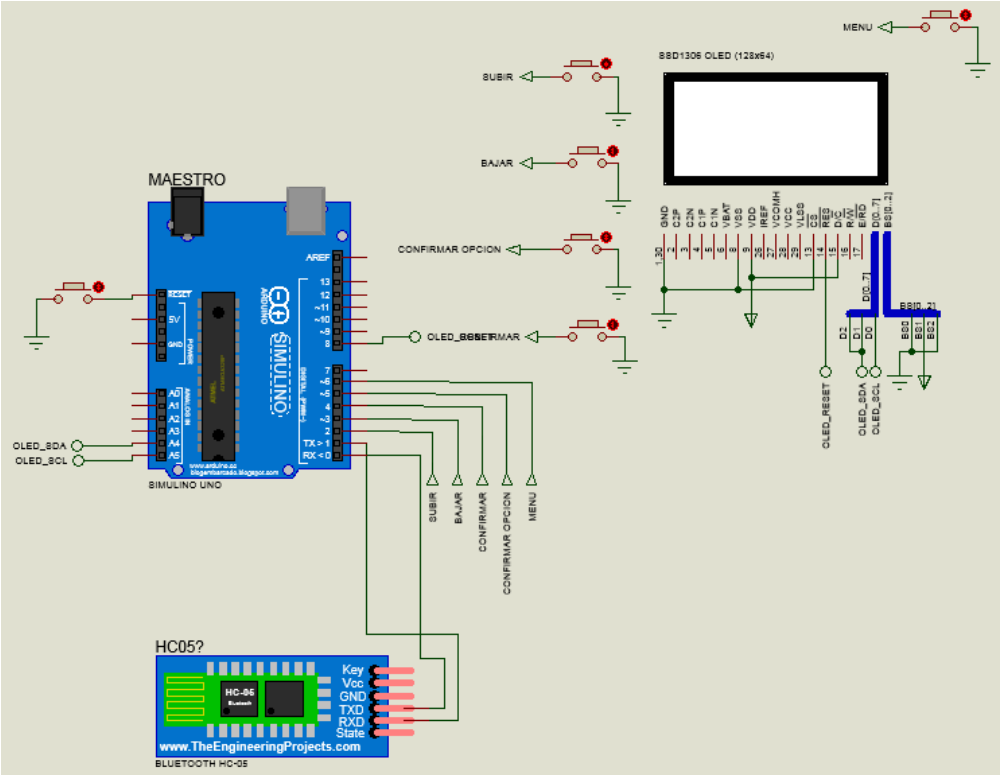


Figura 16: Ensamble completo del Arduino maestro.

Circuito Impreso Arduino Maestro

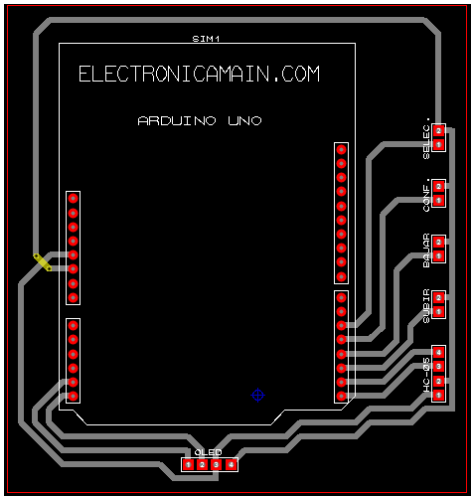
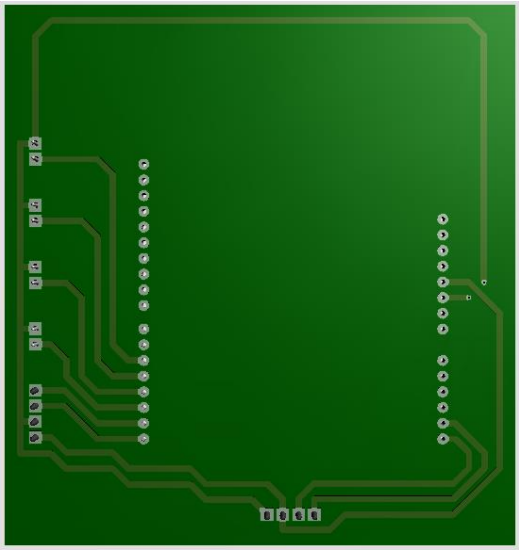


Figura 17: Impreso Arduino Maestro

Tenemos el circuito impreso mostrado en la figura 17 en caso de que vayamos a montarlo o que mandemos a hacer la plaqueta pcb. También hicimos el

modelaje de la plaqueta en 3D para que el usuario que arme el circuito tenga una visión mas amplia de como quedaría una vez completado. Tal y como se muestra en las “figuras 18 y 19”



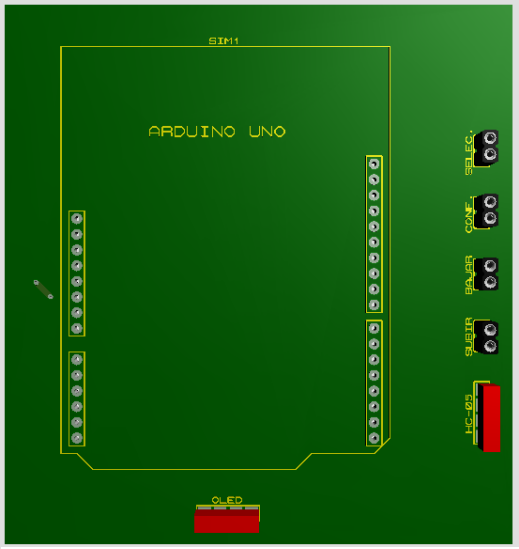


Figura 18: Parte trasera de la placa **Figura 19: Parte delantera de la placa**

Tuvimos que utilizar un Jumper ya que no se nos presento otra forma de completarlo.

Este es el diagrama de flujo por parte del Arduino esclavo

Figura 20: Diagrama de flujo Arduino esclavo

Arduino Esclavo

Una vez que tenemos el maestro procedemos a instalar y programar el Arduino esclavo, como se muestra en la “figura 21”, este mismo nos servirá como un receptor por el que el Arduino maestro mande una palabra escrita por el usuario

y muestre en el lcd que tengamos conectado este mismo Arduino y que pueda alertar de que llego una frase.

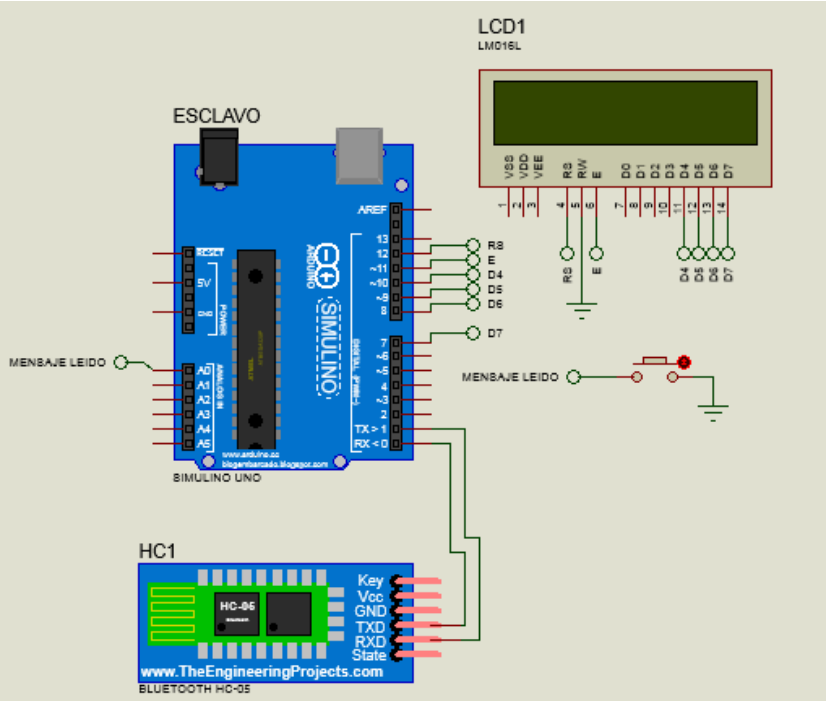


Figura 21: Ensamble del Arduino esclavo

El circuito impreso de Arduino esclavo se muestra en la imagen de la “figura 22”, tenemos tanto el Arduino como el display de 16x2.

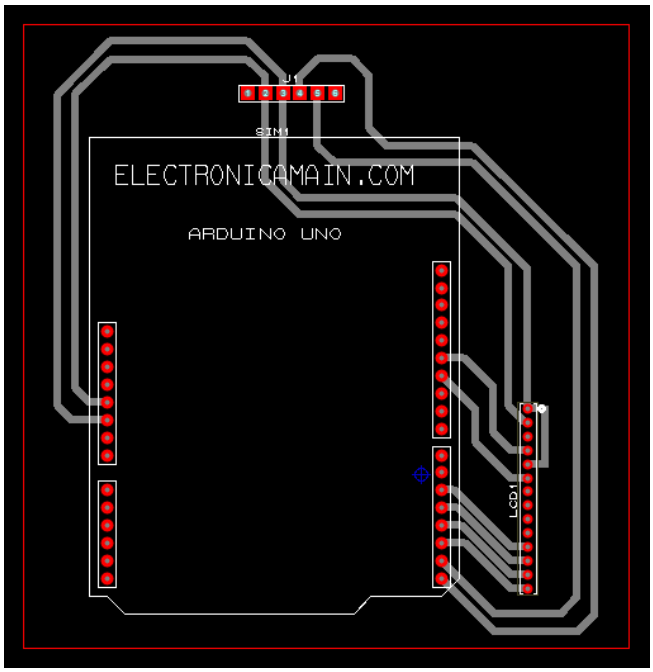


Figura 22: Circuito impreso del Arduino Esclavo

Circuito impreso en 3d del Arduino esclavo.

Para tener una visión de cómo quedaría el circuito ya montado, procedemos a mostrar el circuito en 3d tal y como se muestra en las imágenes de la “figura 23” y ”24”.

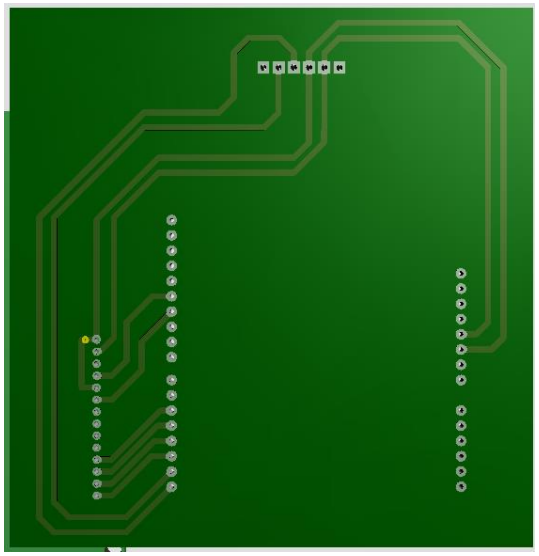


Figura 23: Impreso trasero

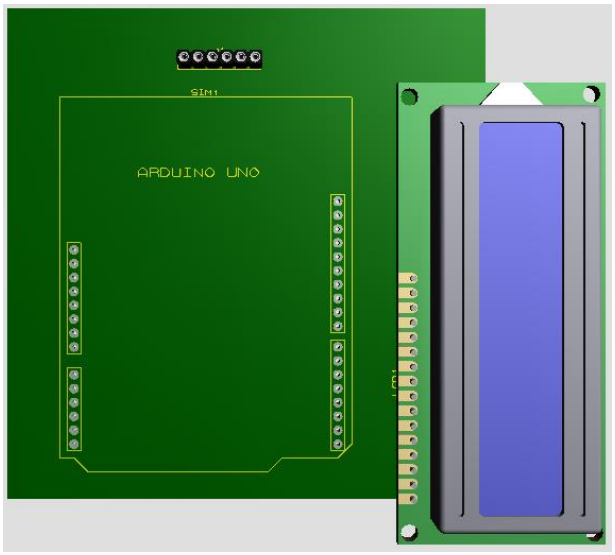


Figura 24: Impreso delantero

Simulación en proteus.

En la figura “figura 25” tenemos imágenes de lo que es la simulación en proteus por parte del Arduino esclavo y en la figura 12.1 tenemos otra imagen de la simulación, pero en esta ya se eligió la opción de confirmar la opción demandada.

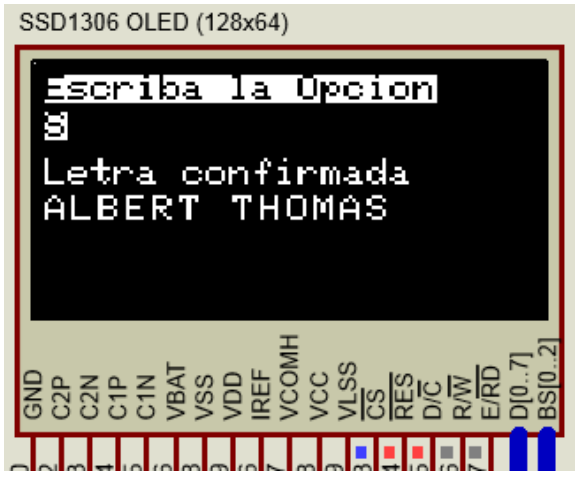


Figura 25: Oled Confirmando Opción

Lo primero que hacer es guardar la opción 1 ya que tenemos un array en el que vamos guardando las palabras que vamos escribiendo. Si ingresamos otra vez a nuestro circuito, ya tendríamos las opciones cargadas por la eeprom. Cuando cargamos las opciones, nos quedaría de la forma en la que se encuentra en la imagen de la “figura 26”

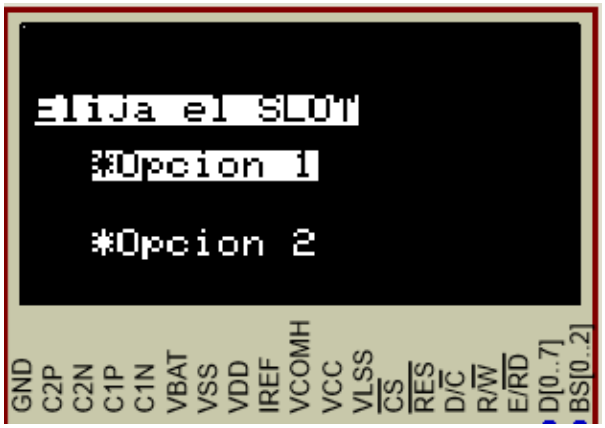
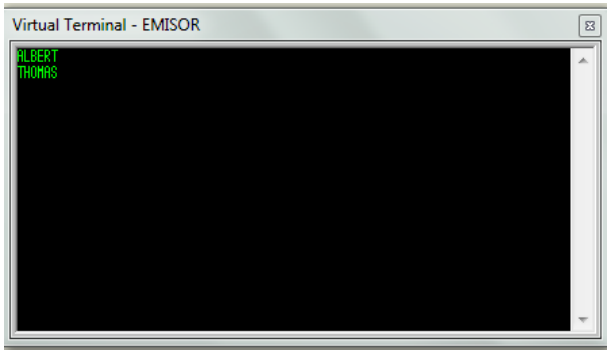


Figura 26: Oled eligiendo slot

Acá, independientemente de la opción que se elija, ya la podemos mostrar tanto por el display como por el serial de cómo se muestra en las imágenes de la “figura 27” y “figura 28”



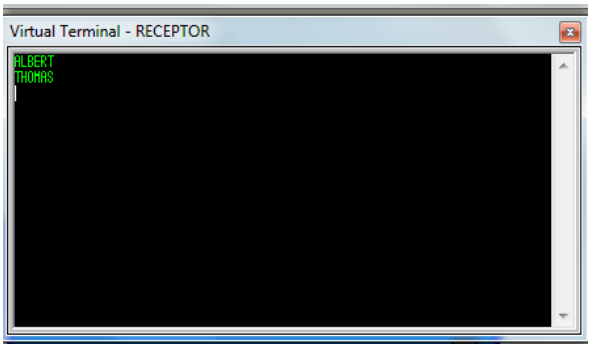


Figura 27: Terminal Emisor

Figura 28: Terminal Receptor

En la “figura 29” se muestra cómo se vería el mensaje enviado a través del Arduino maestro en el Arduino esclavo.

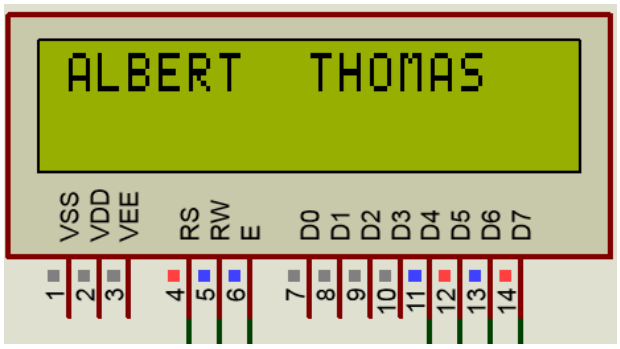


Figura 29: LCD Receptor

Una vez terminado el display del Arduino esclavo como se muestra en la “figura 29“procedemos a conectar en el Arduino maestro el display principal, con este mismo controlaremos el menú principal en donde nuestro cliente podrá insertar las opciones y luego guardarlas, con una interfaz en la que muestra el nombre del colegio. El esquema final sería el que se muestra en la “figura 30”

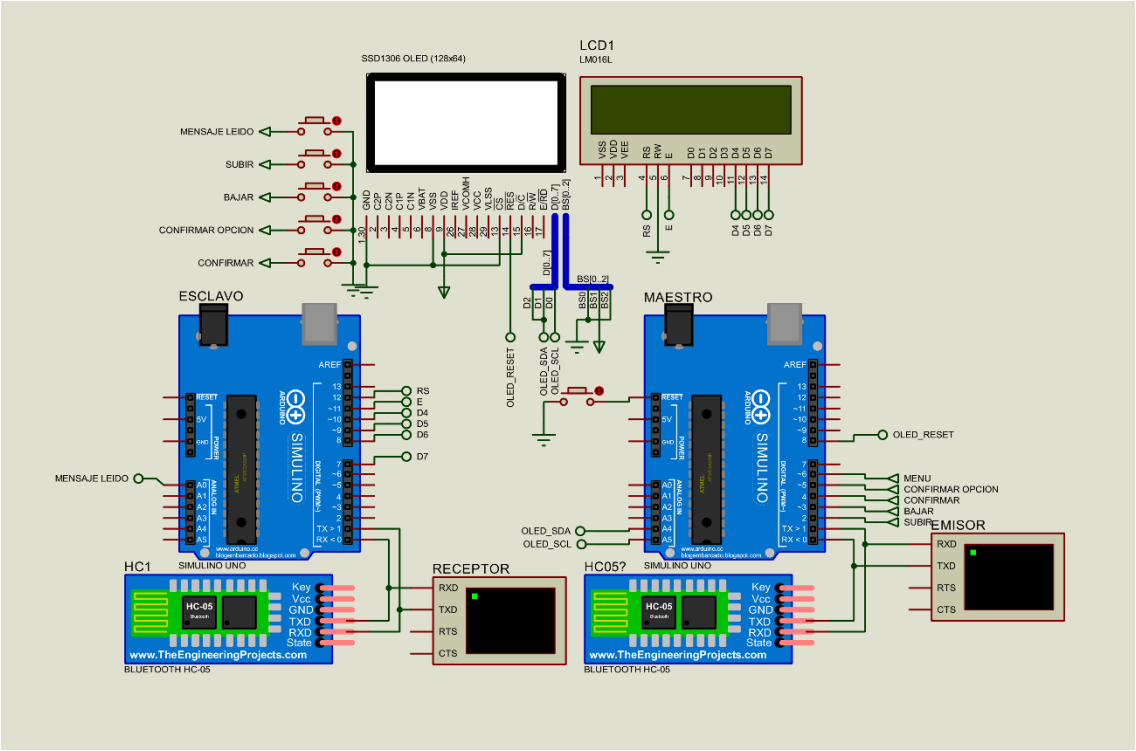


Figura 30: Ensamblaje Final del Proyecto

Conclusión.

En la primera parte del trabajo nos dedicamos al armado básico del Arduino maestro (Los pulsadores más los monitores serial), en este mismo programamos para que el cliente pueda ingresar palabras a su comodidad y mandarlas al menú principal. Por parte del receptor, tenemos un Arduino esclavo el cual se encarga de cargar la frase que fue cargada por el Arduino maestro y mandarla al segundo display de 16x2, en este mismo se mostrara el mensaje enviado por el cliente.

Al finalizar el proyecto, pudimos terminar con todos los objetivos planteados ya que cumple con todos los requerimientos necesarios, para el funcionamiento básico del circuito.

Personal

Como Fuimos separando partes del trabajo

Tuvimos una charla en la que estuvimos optando por la zona en la que más nos cómodo no sintiéramos. Hicimos un diagrama en bloques general y a partir de ahí fuimos dividiendo los bloques por persona, equilibrando el trabajo entre cada integrante del grupo. Luego cada integrante hacia el diagrama individual independientemente de los bloques que hallan tocado.

Como juntamos el trabajo

Al ser programación, cada uno pudo dividir su sector de la forma más legible posible, usando funciones. Al utilizar dos LCDS tuvimos que tocar casi todas las funciones, pero no en gran escala.

Inconvenientes y problemas

Tuvimos varios problemas con el hecho de poder coincidir con el tema de reunirnos para juntar partes del trabajo, ya que cada uno tiene sus actividades o el hecho de estar demasiado ocupado por el tema de las tareas que seguimos haciendo por parte del colegio.

¿Cómo se sintió tener clases virtuales?

A pesar de tener un internet bastante malo, por mi parte, siento que fue lo mejor que me paso en cuanto a organización de tiempos ya que yo soy una persona bastante ocupada. Al estar todo el tiempo en casa y hacer todo por la computadora siento que ya no perdemos tiempo en viajes o en comida para el día y podemos aprovechar el tiempo de mejor manera.

Al ser el sector Arduino, creo yo que no nos cambió mucho ya que los trabajos los hacemos por la pc o portátil.

Que nos gustó y que no nos gustó de las practicas

Me gustó la idea de que nosotros eligiéramos el tema en base a los requisitos de las prácticas y no depender de un trabajo que puede que no nos motive tanto, aunque si tenemos que hacerlo, tenemos que hacerlo, pero al ser nuestra decisión, le ponemos más ganas.

Como Planteamos el Proyecto

Primero, nos juntamos con los chicos del grupo y tratamos de hacer una lluvia de ideas mientras terminábamos trabajos prácticos. Una vez terminado el trabajo en su momento, ya teníamos en mente el proyecto para hacer en las practicas.

Herramientas durante las practicas

Durante las practicas aprendí a utilizar GitHub, (es un controlador de versiones) ya que para trabajos más grandes es bastante bueno ya que nos permite ver el código desde la web y descargarlo en el mismo lugar.

Tareas realizadas de forma individual y grupal

De forma individual: Yo me encargue de que los pulsadores puedan recibir las palabras, que se almacenen en el arreglo y luego se muestren en el puerto serial del Arduino maestro

De forma grupal: Como cada uno ya tenía bien definido la parte que tenía que hacer, las únicas veces que nos juntamos fue para unificar y corregir cosas del mismo código.

Por qué elegimos Arduino

Por mi parte, elegí Arduino porque era la que más se asemejaba a mis gustos como estudiante. Ya que me gusta programar y si puedo hacerlo mientras termino las practicas mucho mejor.

Trabajos Prácticos Realizados

Primero tuvimos que hacer un trabajo de una cerradura en Arduino, para poder conocernos un poco entre los compañeros (Ya que este año no tuvimos la oportunidad) y conocer la modalidad tanto la del informe como la de trabajar en equipo.

Pueden ver los trabajos en:

https://github.com/FabianMartinez1234567/Arduino-Cerradura-.git


TP1.rtf

Mejoras:

- Hacer una api para controlar los pulsadores
- Aumentar la velocidad de lectura de teclas
- Pulir la interfaz del usuario
- Tenes alimentación para ambos arduinos

Antecedentes.

https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

Modelado 3D y empaquetado para la PCB del Arduino Uno para proteus:

https://www.youtube.com/watch?v=tBDHahe3sBQ&t=72s

Funciones del puerto Serial:

https://www.arduino.cc/reference/en/language/functions/communication/serial/

https://www.geekfactory.mx/tienda/radiofrecuencia/hc-05-modulo-bluetooth-maestro-esclavo/

https://www.apuntesdeelectronica.com/microcontroladores/arduino.htm

http://robots-argentina.com.ar/didactica/uso-de-la-eeprom-de-arduino/

https://hetpro-store.com/TUTORIALES/arduino-eeprom/#:~:text=Para%20empezar%2C%20el%20micro-controlador%20en%20el%20Arduino%20tiene,son%20guardados%20aun%20cuando%20la%20tarjeta%20es%20apagada.

https://monarcaelectronica.com.ar/productos/display-oled-1-3-128x64-i2c-ssh1106-arduino-pic-pi-mona/

Repositorio en GitHub: https://github.com/FabianMartinez1234567/Practicas-Botonera