24 de Septiembre

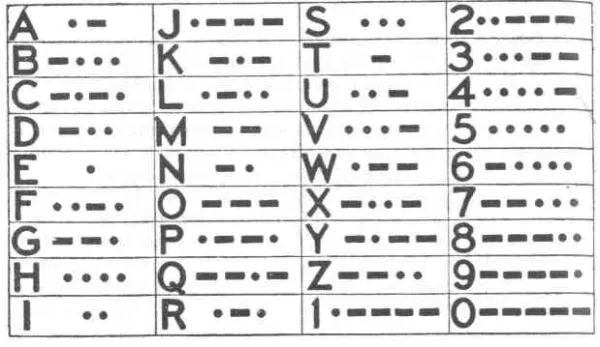
Empezamos investigando sobre el proyecto, ya que nos lo pintaron como el más difícil. Aún estamos en duda sobre si realizarlo o cambiar la idea. En youtube hay varios videos los cuales nos pueden ayudar a lo largo de lo que queda del año, como los pueden ser:

* [codigo morse en Arduino (sonido y simbolos)](https://www.youtube.com/watch?v=lunnM6lPkyo)
* [TRADUCTOR A CODIGO MORSE REALIZADO EN ARDUINO PASO A PASO|Morán](https://www.youtube.com/watch?v=hTC7e5rX1tk)

Plantee el problema y me hice una idea de como traducir de ambos lados(morse a texto y viceversa).

Mi idea es hacer uso de un botón y las señales que envía en conjunto al monitor sería para enviarle datos al arduino.

Utilicé la siguiente imagen de google para formar los vectores que componen la traducción de números y letras, para en un futuro poder usarlos como traductores texto a morse así como de morse a texto:



Averigüe sobre la función “pulseIn”. Esta permite calcular el tiempo en microsegundos que tarda una variable booleana(T/F, HIGH/LOW, 0/1, etc) en cambiar su valor. Esta función cuenta con un radio funcional entre 10 microsegundos y 3 minutos.

Mi idea es usarla para leer un pulsador y dependiendo del tiempo que se mantenga apretado el pulsador, el sistema lo interprete como un punto, un guión o un espacio.

Dicha función se escribe así:

| pulseIn(pin, valor) |
| --- |

Donde pin es el pin que se desea calcular el tiempo que tarda en variar y valor es el estado al que se quiere variar (si el valor en los parámetros es HIGH, la función esperará a que el pin especificado valga HIGH y de allí cuando cambie a LOW, empezará a contar el tiempo que tarda en volver a cambiar a HIGH).

Otra función nueva que aprendí es la de Serial.read. Esta permite ingresar datos por el monitor serial y guardarlos en una variable.

Esto es muy útil para poder enviar textos al arduino para posteriormente traducirlos con ayuda de un buzzer en código morse:

Como se escribe la función

Serial.read();

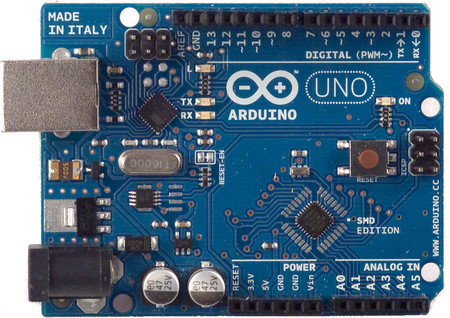
Guardamos en una variable y lo mostramos por la consola

| String texto = Serial.read(); Serial.println(letra); |
| --- |

Componentes a utilizar:

**Arduino:**

Es la pieza más importante para cualquier proyecto. Es el encargado de enviar y recibir señales digitales y analógicas a todos los componentes que implica el trabajo. A el se conectan todos los cables que a su vez están conectados a los diferentes componentes que en conjunto permiten el funcionamiento del código.



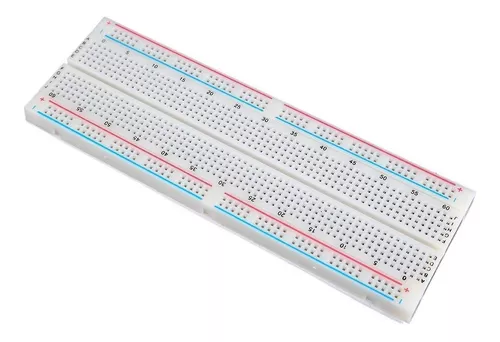
**Pulsador:**

Mediante este nos comunicaremos con el arduino, enviando señales en código morse en forma de pulsaciones que serán leídas con pulseIn para interpretar el dato que queremos enviar, para luego con unos vectores transformarlo en letra y formar un texto.



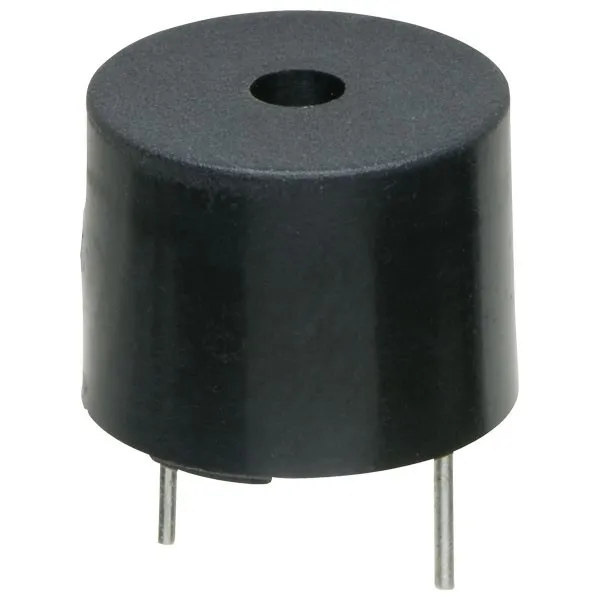
**Protoboard:**

La protoboard es otra parte fundamental para el funcionamiento del sistema. Ya que se encarga de enviar señales eléctricas mediante los cables para así ahorrar espacio y cables. En sus ranuras irán conectados todos los cables de los diferentes componentes(incluidos el GND y el 5V del arduino), mediante los cuales trasladará la energía que los componentes envían o recibirán y así de esta manera hacer funcionar el sistema satisfactoriamente.



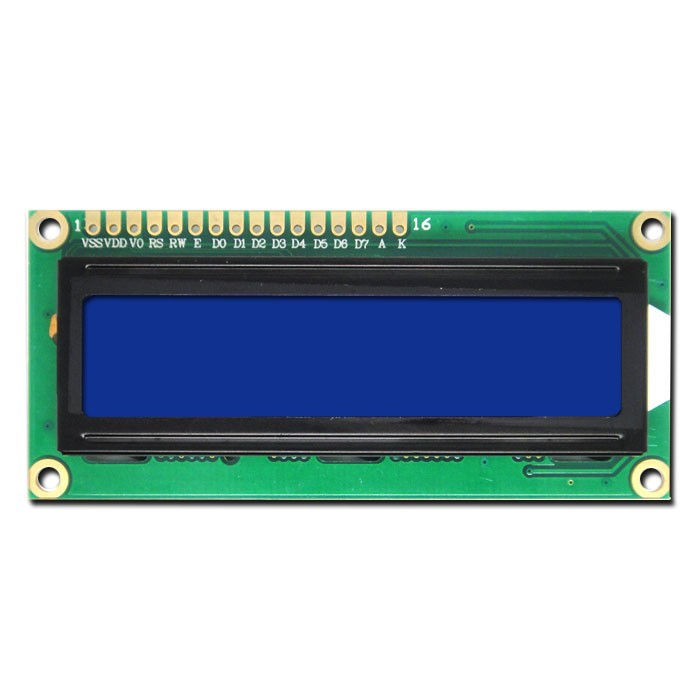
**Buzzer(opcional):**

Instrumento que emite sonidos en base a lo especificado. Cada vez que se traduzca una letra del texto enviado este hará se prenderá por más o menos tiempo, representando así un punto o un guión. Esta secuencia se repetirá hasta terminar la letra.



**Pantalla LCD:**

La pantalla LCD(Liquid Crystal Device), permite mostrar información por su propia pantalla. Dentro del código y su funcionamiento, esta será utilizada para mostrar las letras traducidas luego del procedimiento del código, por ejemplo, si se ingresa “.-” que su equivalente en nuestro alfabeto es “A”, se mostrará por esta pantalla dicha letra.

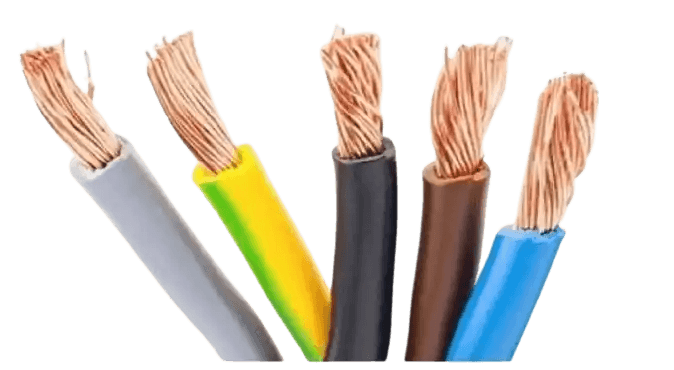


**Cableado:**

Gracias a este podemos garantizar el funcionamiento del sistema. Ya que es el encargado de

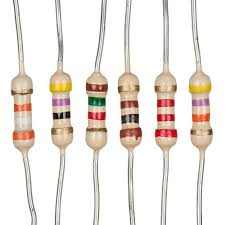
transmitir la electricidad para el correcto funcionamiento de los componentes. Mediante estos se

traslada la información que recibirán o que envían los diversos componentes.



**Resistencias:**

Permiten controlar el paso de electricidad por los cables para no enviar exceso a ciertos componentes. Por ejemplo: un LED no soporta una entrada de 5V(la energía base que envía el arduino por uno de sus pines), por lo que si conectamos un LED a esta fuente de energía se quemaría. Por eso conectamos la resistencia, que atenúa el paso de energía y evita problemas con los componentes



**Bluetooth:**

Gracias a esta herramienta podemos hacer que dos arduinos se comuniquen entre sí, enviando

información y cumpliendo una parte del proyecto. Todavía no se como se hace la conexión bluetooth ni qué componentes serán utilizados, pero el objetivo que cumplirán sera de enviar señales digitales y/o analógicas entre arduinos, para sí detectar el tipo de dato y el contenido de este para tratarlo de una manera u otra.



**Led:**

Cumple la misma función del buzzer solo que visual.

Por cada punto o guión que interpreta el sistema, el led se prenderá por más o menos tiempo.



8 de Octubre

En el día de hoy empecé sugiriendo la creación de un documento compartido para comunicarnos más fácilmente y dividirnos las tareas. Por ahora mi parte es la de hacer que el Arduino interprete las pulsaciones de un botón y las traduzca a código morse.

Hice la siguiente estructura de código:

| while(true){  int codigo = pulseIn(boton,HIGH);  if (codigo > 1 && codigo < 3000){    }  else if(codigo > 3000 && codigo < 5000){    }  else if(codigo > 5000){    }  else{  break;  }  } |
| --- |

Los valores no son los idóneos (o eso creo) pero me sirven de referencia. Mi idea es que dependiendo de lo que se cumpla(código es el tiempo que se mantiene presionado el pulsador) se prenda un buzzer y suene más o menos tiempo para traducir el morse. Si no se aprieta el pulsador

el ciclo se rompe y se imprime la palabra.

**Primer error!!!:**

Por alguna razón que no logro comprender, el pulseIn funciona mal y devuelve valores negativos(aunque también positivos). Consulté el error con copilot y lo busqué en foros de google pero no encontré nada al respecto. El profe nunca tuvo ese error por lo que no sabe cómo ayudarme. Seguiré buscando hasta solucionarlo.

Intenté usar el código que me brindó copilot, que es para aumentar su timeout(su delay según entendí). No cambió mucho.

Estoy considerando la opción de multiplicar por -1 al valor del pulseIn si es negativo. El problema es que devuelve un valor pequeño que no se parece al resto.

El profe me enseñó un nuevo tipo de dato, el cual es el “unsigned int”. La función de este es la de quitarle el signo al número, aunque me siguen devolviendo números raros.

Ya termino la clase, en la semana trabajaré en mi casa. No hice mucho por hoy y soy consciente que cada vez se acerca más la fecha final. De los 5 que somos solo trabajamos 2 y fue una producción muy ralentizada. Espero que para la semana que viene trabajemos mejor.

14 de octubre

Adelanté un poco mi código. Ahora en mi casa con un mejor wifi aproveche para ver el error de la semana pasada. Sigue igual(aunque hoy no está andando muy bien mi internet). Ignorando ese error y usando la función “unsigned int” arme una estructura del código que se vería de la siguiente manera:

| while(true){  unsigned int valor = pulseIn(boton, HIGH);  valor = valor / 1000000;  if (valor > 0 && valor < 0.7){  char caracter = '.';  digitalWrite(buz, HIGH);  delay(250);  digitalWrite(buz, LOW); |
| --- |

El código continúa pero ese es el patrón. Lo que quise hacer es que dependiendo del valor el buzzer suene más o menos tiempo, o directamente no suene. Por ahora va más o menos bien. Tampoco tengo pensado progresar mucho más hoy ya que tengo un par de dudas que quiero preguntarle al profe mañana, como por ejemplo que el buzzer no suena hasta que toque nuevamente el botón. Por ahora dejaré de lado el error de los valores negativos(ya que aunque los haga positivos siguen siendo valores diferentes a los que me debería devolver) pero se que tarde o temprano tendré que arreglarlo.

15 de octubre

Encontré la siguiente página. Me sirvió para guiarme y darme una idea de que hacer, no me voy a copiar tal cual si no que lo estuve tratando de entender a fondo así lo puedo hacer a mi manera(el link aparece como no válido pero si lo pegas en tu navegador te redirecciona a la página):

[www.https://blogs.etsii.urjc.es/dseytr/traductor-bidireccional-espanol-morse/](about:blank)

Investigué en google y voy a hacer uso de la función length, que se escribe así:

| string.length(); |
| --- |

Donde string es el nombre de la variable tipo string. Esta función devuelve la cantidad de caracteres que tiene dicha palabra. Ej: String palabra = “hola”; palabra.length(); devuelve 4.

Avancé bastante más. Sigo aprendiendo cosas nuevas, como la de que puedo tratar a un string como un vector, donde cada una de sus posiciones es cada una de sus letras. Esto lo estoy usando para determinar qué letra o número es el código que ingresé en morse. El algoritmo es el siguiente:

recorro el String mediante un ciclo for e igualo la posición a un caracter (‘/’), si la posición actual es diferente, añade ese caracter a un string y repite el ciclo. En caso de ser igual agarra el String que fue formando y mediante otro algoritmo detecta que letra en nuestro alfabeto es su equivalente. Luego de eso imprime la letra equivalente en la pantalla LCD, y por último limpia el String para volver a empezar con la formación de la siguiente letra. El código funciona de la siguiente manera:

| for(int i = 0; i < longPalabra; i++){  if(palabra[i] != '/'){  letra = letra + palabra[i];  }else if (palabra[i] == '/'){  for(int i = 0; i < 26; i++){  if(letra == letrasMor[i]){  LCD.print(letras[i]);  contPos++;  }}  for(int i = 0; i < 10; i++){  if(letra == numerosMor[i]){  LCD.print(numeros[i]);  contPos++;  }}  letra = "";  }else if((i + 1 < longPalabra) && (palabra[i] == '/' && palabra[i+1] == '/'){  LCD.print(" ")  } |
| --- |

Donde mi objetivo es que una vez tenga toda la secuencia traducida en morse, un ciclo for recorra la palabra y vaya separando las secuencias de puntos y líneas entre las barras y las convierta en números o letras mediante el uso de vectores. Aún no probé esta porción de código, por lo que no sé si funciona(aunque todavía me faltan partes de este).

Runee el código. Me tiró varios errores;

1- Me faltó definir el tipo de dato para la variable contPos, error boludo que por suerte me di cuenta rápido.

2- Otro error boludo, me faltó definir la variable global de caracter.

3- Me faltó definir el tipo de dato de la variable longPalabra.

4- Me falto un ‘;’.

creo que esos eran los únicos errores que tiraba el código. Todos errores sin gravedad ya que son fáciles de arreglar. Todavía no probé si el código hace lo que quiero que haga, pero por lo menos no tira error.

Ya es todo por hoy. Día bastante productivo la verdad, siento que casi tengo terminada mi parte. La semana que viene armaré el circuito en arduino físico y probaré si todo anda como corresponde. Los valores negativos creo que ya no aparecen más, aunque cada tanto si devuelve un valor medio extraño. Esto lo solucioné con la ayuda del profe, ya que en donde tenía el pulseIn tenía configurado para que leyera cuando el botón pasara de valer high a valer low, cuando yo lo tenía conectado en pullup(lo que quiere decir que de por sí me devolvía un valor low). Cambie esto y anda mejor.

Conforme pasa el tiempo me voy dando cuenta que este trabajo lo tendremos que hacer entre 2, ya que 3 de mis 5 compañeros no hicieron nada hasta ahora más que estar boludeando con el celular o verme trabajar. No se como va mi otra compañera que está trabajando, pero espero que vaya bien. Cada día veo más cerca el fin de año y la verdad si tengo miedo por la nota de este proyecto.

22 de octubre

Hoy empecé con el arduino físico. Hicimos las conexiones necesarias junto a un compañero y probamos. Al principio no nos funcionaba porque estaban mal conectados el positivo y negativo. Luego de arreglarlo funcionó lo que teníamos conectado. Tuvimos que llamar al profe para que nos enseñe cómo subir el código al arduino físico(conectar mediante un USB el arduino a la computadora, luego abrir la aplicación de arduino, cargar el código y tocar el botón “subir”). Mientras escribo esto recordé que no me fije si el problema de los negativos continúa aún en el físico, lo voy a probar ahora mismo.

Ya probé y no cambio nada. El arduino sigue devolviendo números negativos cada tanto por el monitor serial. Esto no sería tanto un problema si no porque me devuelve valores bastante menores/mayores a los que me debería devolver, convirtiendo ocasionalmente lo que sería un punto en un guión o viceversa. Ahora estamos en recreo, cuando venga el profe le vuelvo a preguntar a ver si me ayuda o esta vez logra comprender qué pasa con este error.

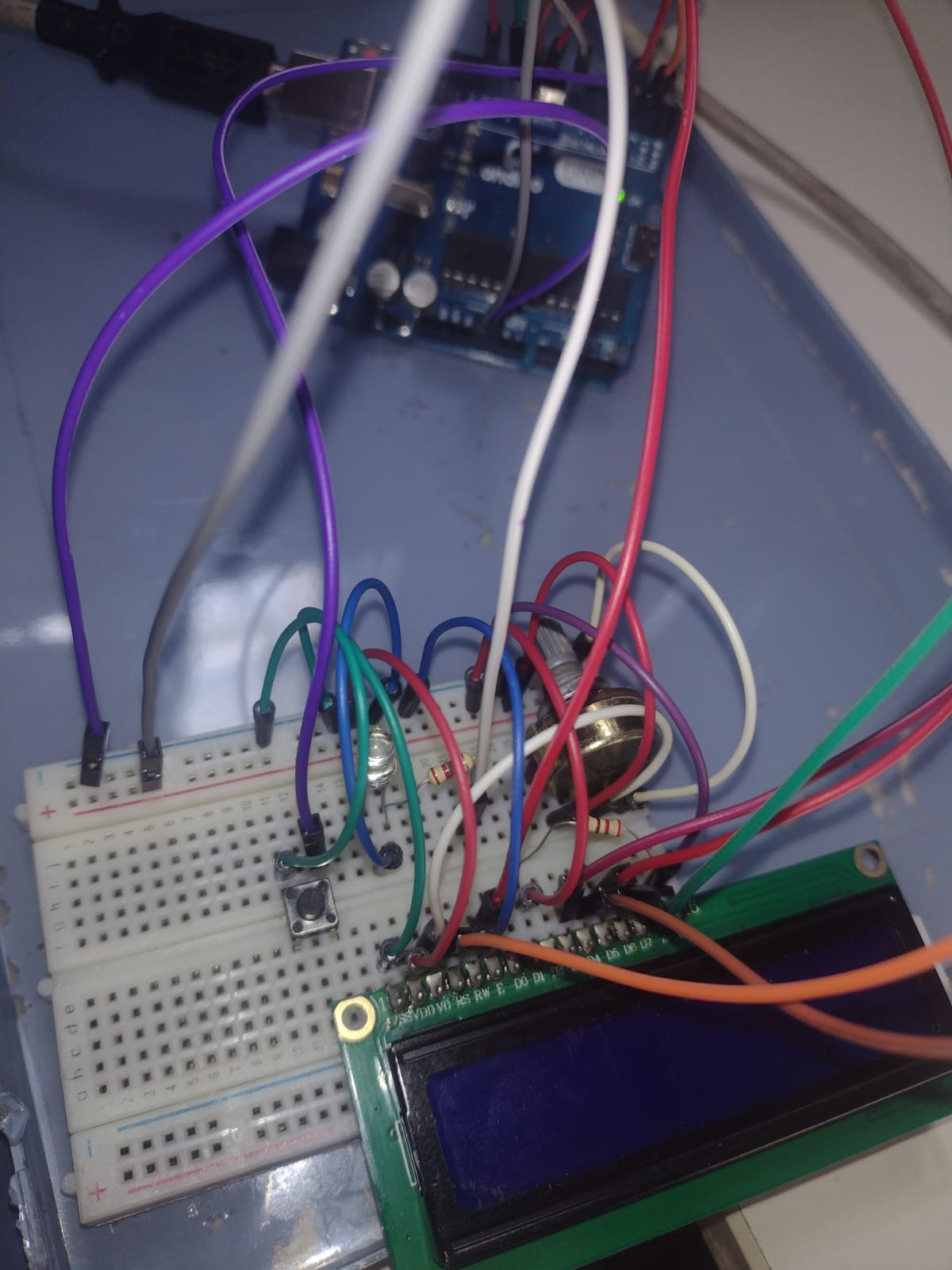
Actualización:

Llamé al profe, se fijó en el proyecto de Morse del año pasado y encontró la solución. Resulta que la función de pulseIn devuelve valores con coma, y como lo tenía como int (unsigned int para evitar valores negativos) estaba forzando a la máquina a convertir decimales en enteros, por

lo que los convertía en números negativos. Al asignarle el tipo de dato como float, evitamos este error y ya devuelve todos números positivos y bonitos.

Terminé el día con un no muy gran avance, hicimos las conexiones necesarias para probar todo y funciona a medias. Aún hay muchos errores que resolver, como uno que no lee bien el botón(no lo lee por mucho tiempo, por lo que solo puedo generar puntos). Espero que la semana que viene salga todo mejor.

Es bastante difícil conectar todo por el espacio reducido, sumado a que hay muchos cables rotos y las conexiones no son muy firmes(se desconectan solas fácilmente). De todas formas sirvió de experiencia y para tener práctica y mayor fluidez para cuando llegue el momento de juntarlo todo. Las conexiones del arduino quedaron de la siguiente manera:



29 de octubre

Hoy empecé el día solucionando un problema que tenía en tinkercad. Este consistía en que no me prendía la pantalla LCD, ya que tenía varios errores.

Para empezar lo tenía mal configurado. Ya que había puesto mal los pines al momento de escribir el código de configuración de pines. Este código pide 6 pines: el RS(register select), Enable, DB4, DB5, DB6 y DB7. Yo tenía el siguiente código:

| #include <LiquidCrystal.h> LiquidCrystal LCD(2,3,9,10,11,12); |
| --- |

La cuestión es que el DB7 estaba conectado al pin número 13 y yo lo tenía configurado para el pin 12. Un error boludo, pero me pasan cosas como estas todo el tiempo.

Otro error fue uno relacionado con los “else if”. Tenía el código lleno de else if por todas partes. Lo que hizo que el código no entrará a ninguno de estos, porque estaba mal planteado, ya que por distracción coloqué muchos else if y muy pocos if.

| for(int i = 0; i < longPalabra; i++){  if(palabra[i] != '/'){  letra = letra + palabra[i];  }  else if (palabra[i] == '/'){  for(int i = 0; i < 26; i++){  if(letra == letrasMor[i]){  LCD.print(letras[i]);  contPos++;  }  } |
| --- |

Este es un ejemplo, así como el de abajo. Analizándolo ahora no entiendo muy bien porque tiraba error, pero creo que es por que para entrar en el else if, el if debe devolver un valor negativo, y esto no estaba sucediendo. Lo importante es que logré solucionarlo, así que no me voy a quemar la cabeza intentando comprenderlo, y simplemente seguiré con el código para terminarlo cuanto antes:

| if(i == 9 && letra != numerosMor[i]){  LCD.print("?");  }}  letra = "";  }else if((i + 1 < longPalabra) && (palabra[i] == '/' && palabra[i+1] == '/')){} |
| --- |

Actualización de los componentes:

Añadimos un led y quitamos el buzzer. Esto debido a que al momento de trabajar en físico el buzzer podría resultar molesto y/o estresante para los compañeros(según el profe para que no nos maten). De todas formas, ambos cumplen la misma función.

Ahora el LCD funciona bien. Lo último que me gustaría hacer es que cuando no encuentre un caracter escriba un signo de interrogativo (“?”), para que el usuario pueda saber cuando ingresó mal un caracter. Parece que hoy no hice mucho, pero me sirvió para adelantar mi parte y quitarme problemas futuros de encima(tuve que quitar una parte del código por que si no ocupaba mucho espacio):

for (int j = 0; j < 26; j++) {  
 if (letra == letrasMor[j]) {  
 if (contPos < 16) {  
 LCD.setCursor(contPos, 0);  
 LCD.print(letras[j]);  
 } else {  
 LCD.setCursor(contPos - 16, 1);  
 LCD.print(letras[j]);}  
 letra = "";  
 contPos++;  
 }}

4 de noviembre

Hoy me encargué de perfeccionar y corregir los errores que me faltaban de antes. Lo primero que hice fue modificar el código, para conseguir una estructura funcional.

Para empezar cambie el funcionamiento, ya que antes el código funcionaba mediante un ciclo while y ahora funciona por un ciclo for. Gracias a esto, ahora puedo determinar cuando finaliza el código y con se repita infinitamente. También puedo detectar con mayor facilidad por el ciclo en el que se encuentra el sistema y para no tener que poner un break en un if ubicado en un lugar determinado del código para finalizar el ciclo y poder continuar con el resto del funcionamiento de este..

| for (int i = 0; i < longPalabra; i++) {  if (palabra[i] != '/') {  letra += palabra[i];  }   if (palabra[i] == '/' || i + 1 == longPalabra) {  if (contPos < 16) {  LCD.setCursor(contPos, 0);  } else {  LCD.setCursor(contPos - 16, 1);  }  contPos++; |
| --- |

Hice unas líneas de código que me permiten empezar nuevamente desde la primera posición de la pantalla LCD en caso de que la frase ingresada traducida exceda los 32 caracteres (incluyendo espacios), es decir, que sobrepase los espacios disponibles de la pantalla LCD, así como cuando termine de escribir la palabra por completo vuelva a empezar. Eso está en el final del código. Aunque el if va dentro del ciclo for que engloba todo, mientras que las otras líneas de código están fuera del for. Ya que son las encargadas de limpiar la pantalla y volver a escribir la palabra una vez se termine de escribir.

| if(contPos > 31){  delay(3000);  LCD.clear();  contPos = 0;  }} }  contPos = 0;  delay(3000);  LCD.clear(); |
| --- |

Por último añadí más cosas como una nueva variable (encontrado) que permite ver si la secuencia de caracteres existe en morse o no:

| bool encontrado = false;  for (int j = 0; j < 26; j++) {  if (letra == letrasMor[j]) {  LCD.print(letras[j]);  encontrado = true;  break;  }  } |
| --- |

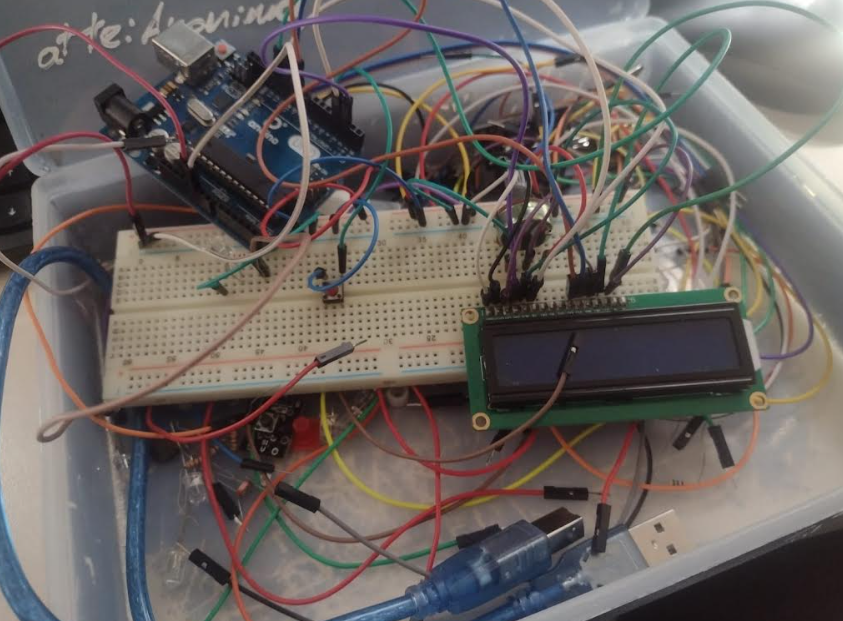
Una vez teniendo su respectivo valor la variable “encontrado”. El sistema revisa su valor (verdadero o falso) y el de la letra(que su valor se actualiza por cada caracter registrado) y si es falso(si no se encuentra la traducción en morse a la secuencia de caracteres) imprime un signo de pregunta:

| if (!encontrado && letra != "") {  LCD.print('?');  } |
| --- |

Probablemente mañana en clase o si no en mi casa cuando tenga tiempo libre haré todo más prolijo y lo separaré en funciones, pero esto es todo por hoy.

5 de noviembre

Volví a intentarlo en el físico. Estuve todo el día tratando de solucionar errores relacionados a esto. Aparentemente la pantalla LCD estaba rota o andaba mal, ya que no imprimía por la pantalla lo que le pedía imprimir(parece ser que no andaban los LEDS). Por otra parte, otro error que no me enfoqué en solucionar fue que el botón devuelve únicamente valores bajos y no devuelve nada si lo mantengo apretado, al igual que a veces no funciona(lo toco y no pasa nada):

Hoy hice poco y nada. Grán parte del día me la pase esperando a que mi compañero hiciera las conexiones(no las hacía yo de primeras por que quería ver si hacía algo, ya que no hizo nada hasta ahora en todo lo que vamos de proyecto) pero siempre las hacía mal y tenía que intervenir para que estas funcionen bien, luego le decía que revisará y tardaba bastante tiempo más. Así quedaron las conexiones que hicimos hoy:

12 de noviembre

Hoy, como ya terminé(o casi) mi parte del proyecto, comencé a ayudar con lo restante. Hice una parte de la función de traducir por texto de morse a texto normal (o por lo menos así lo interpreté yo). Utilicé la variable Serial.read para leer texto y luego traducirlo(esto todavía no lo hice, puede que lo haga en mi casa o la semana que viene). Respecto a esto, hoy me enteré que el monitor serial no lee string cuando se le ingresa texto por un Serial.read, si no que solo lee números y caracteres, por lo que tuve que usar una función de la cual tenía una base por que otra persona de mi grupo se había encargado de averiguar y preguntar al profe:

| void loop() {    if( Serial.available() )  {   String g = "";  while( Serial.read() != -1){  g = g + (char)Serial.read();  }    Serial.println( g );  }  } |
| --- |

En un principio, ese código no funcionaba, ya que devolvía un caracter extraño(ÿ) pero lo pude arreglar modificando algunas cosas: En vez de utilizar el fragmento de código “(char)Serial.read();

if (Serial.available()) {   
 String caracter = "";  
   
 while (Serial.available()) {   
 char letra = Serial.read();  
 caracter += letra;  
 }  
}

Ahora si funcionaba bien el código y pude empezar con lo otro. Intenté seguir el mismo algoritmo que en la otra parte del código que había hecho, aunque también empecé a separar el código en funciones para acortarlo, y agilizar su funcionalidad.

Por otro lado, más temprano probé el otro código, que antes andaba raro y ahora ando mejor, ya que este me leyó las “/” y pude probarlo con más fluidez. Hoy no toqué el arduino físico pero aún así logré un progreso en el proyecto(y más teniendo en cuenta que soy el único que hace algo). Quedan pocas semanas de clase y no se si lo llegue a terminar, espero que aunque sea mis compañeros me ayuden a hacer el powerpoint para la presentación. Así quedaron algunas de las funciones:

Esta función es el código que pase arriba, pero más ordenado y por si lo tuviese que usar más

veces.

Por otro lado, la función de abajo, cumple la función de revisar línea por línea si la secuencia de caracteres(en morse) coincide con la posición del vector que ingrese(sería primero el de números), lo que hace que realice varias acciones y devuelva un valor(encontrado), que me permite repetir la función con los números o comprobar si se encontró la letra(o número) en los vectores, y en caso que no imprimir un “?”(esto mediante otra parte del código que no mostré). Ahora sí, eso es todo lo que hice hoy, no fue la gran cosa pero si fue un buen progreso. Espero llegar a tiempo para presentarlo y que mis compañeros hagan algo más que sentarse a mirar o jugar con el celular.

Esta es la otra función que hice y de la que estuve hablando arriba. Como dije, es solo para ordenar y no metí nada nuevo:

| bool buscar(String string,int hastaDonde,String vectorComparar[],String vectorImprimir[]){  bool encontrado = false;  for(int n = 0; n < hastaDonde; n++){  if(string = vectorComparar[n]){  LCD.print(vectorImprimir[n]);  encontrado = true;  break;    }  }  return encontrado; } |
| --- |

19 de noviembre

Cuenta regresiva, estamos a pocas semanas de la fecha límite del proyecto y lo veo difícil.

No logro encontrar donde, pero el código anda mal en algún lado. Cuando ingreso “.-”(que es equivalente a “A”), me muestra por la pantalla LCD “ET”(porque “.” es equivalente a “E” y “-” es equivalente a “T”). Esto me da a entender que está leyendo los caracteres de manera individual. Seguiré buscando la solución.

Llamé al profe, no me quiso decir por qué (tampoco leyó mucho el código), pero al ver el problema intuyó que era un problema del recorrido.

Estuve poniendo prints por todos lados pero no encontré nada, hasta que puse uno en la función de lectura(que está más arriba) y noté que se duplicaban algunos caracteres. Llamé al profe nuevamente y me dijo que posiblemente eso se debía a los caracteres de escape, que por lo que observé son los también llamados caracteres especiales(o eso creo). Los caracteres de escape, según lo que encontré en internet son usados para cambiar la información de un valor(por ejemplo, en algunos lenguajes, “a” no es lo mismo que “\a”). Sinceramente no entiendo por que se duplican, pero con ayuda del profe y buscando en la página oficial de arduino, encontré una nueva función que me puede ayudar:

la función sirve para eliminar uno o varios caracteres de un string, y se escribe de la siguiente manera:

| string.remove(index) |
| --- |

Hay un parámetro más que se le puede pasar a esta función, pero no termino de comprender para qué se usa así que no lo voy a detallar.

Implementé esta función de la siguiente manera en mi función donde tenía el error:

| String lectura(String a){  while (Serial.available()) {   char caracter = Serial.read();  a += caracter;  if(caracter == '/'){  largo = a.length();  a.remove(largo -1);  }  }  return a; } |
| --- |

El problema de esto, es que elimina la barra directamente, y no la duplica tampoco. Por lo que probé con el mismo código solo que cambie el “largo - 1” de la función “a.remove” a simplemente “largo”. Luego de hacer esto, no solo se duplica la barra, si no que también se duplican puntos o líneas (dependiendo de lo ingresado, pero no siguen un patrón específico).

Volví a llamar al profe, y no pudo encontrar en donde fallaba, me recomendó usar inteligencia artificial, y luego de insistirle con varias preguntas logré dar con el problema.

Resultó ser algo tan estúpido como el tener que poner un delay. Por culpa del sistema y de que todo iba muy rápido, se sobreponen los datos y se juntaban 2 en uno solo. Esto lo pude solucionar con un simple delay de 10 microsegundos, evitando así los llamados micro rebotes y logrando estabilizar el código.

| String lectura() {  String a = "";  while (Serial.available()) {   char caracter = Serial.read();  a += caracter;  delay(10);  }  return a; } |
| --- |

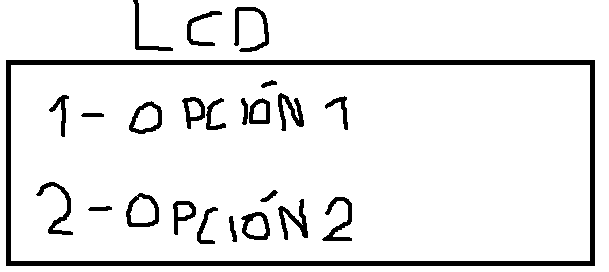
Una vez arreglado esto, el código estaba hecho y lo pasé todo a una función llamada “morseAAlfabeto”. Aparte de esto, me falta una parte en la que en vez de ingresar texto en morse, lo ingreso en nuestro alfabeto y lo traduce mediante tildaciones del LED. Luego me gustaría hacer una especie de menú, donde de a elegir el método de traducción y por último, obviamente unir los dos códigos.

Si hay algo que me da miedo, es la parte del bluetooth, ya que quedan apenas 2 semanas antes de entregarlo y probablemente lo tengamos que resolver la semana que viene, lo que quiere decir que tendré que terminarlo en mi casa. No tengo idea ni se como funciona eso, ya que nunca vi nada ni lo usé, aunque por suerte el profe dijo que nos va a explicar.

24 de noviembre

Mi objetivo de hoy, es terminar todo el código y que funcione bien(aunque si no lo logro hoy, también podría hacer cosas mañana). Ya que ninguno de mis compañeros hizo nada durante todo el proyecto(excepto uno que se hizo la parte del código que traduce el texto ingresado en nuestro alfabeto al alfabeto morse mediante pitidos de un buzzer y luces de un led, el cual no usaré por que no me gustó mucho como funciona y se nota que usó inteligencia artificial para lograrlo y prefiero usar funciones y algoritmos que pueda comprender para así también facilitarme la presentación del proyecto)(aunque yo también utilicé algunas veces IA, pero solo para quitarme algunas dudas o para ayudarme a encontrar un error/ arreglar el funcionamiento del código).

Lo primero que voy a hacer es un menú que permita al usuario elegir la opción de traducción que quiera. Aquí un boceto de mi idea:



Donde la opción 1 vendría a ser el uso del botón, mientras que la 2 vendría a ser el uso del monitor serial(en caso de elegir la 2, daría a elegir otras 2 opciones de si traducir de nuestro alfabeto a morse o al revés).

Mi idea es hacerlo funcionar con varios Serial.read para detectar el ingreso y de ahí poner varios if para dependiendo del ingreso llame a una función u otra. Dichas funciones estarán explicadas más abajo, detallando su funcionamiento y la función que cumplen. Para terminar llamé la función dentro del loop y comencé con el resto.

| LCD.setCursor(0,0);  LCD.print("1 - Boton");  LCD.setCursor(0,1);  LCD.print("2 - Texto");  if (Serial.available()) {  String opcion = lectura();  if(opcion == "1"){  }  else if(opcion == "2"){  }  else{  Serial.println("Opción no válida");  } |
| --- |

Empecé probando ese código(no tiene nada dentro del if ni del else if ya que las funciones todavía no están creadas). A primera vista parecería que no hay ningún error, pero al estar este código dentro de la función loop (que repite el código infinitas veces), el clear me molestaba ya que no se terminaba de escribir el texto y ya se borraba. Por lo que probé ponerlo dentro de la función setup (que se ejecuta una sola vez al iniciar el código).

El resultado de poner el código en el setup, fue que al ejecutarse una sola vez al iniciar el código, no daba tiempo a ingresar datos por el monitor serial. Esto debido a la velocidad a la que se ejecuta el setup, porque leía todo el código y como no daba el tiempo para ingresar un valor

por el monitor serial pasaba al loop directamente. Seguiré pensando cómo solucionarlo.

Actualización:

encontré una solución a mi problema. Mi idea fue hacer un while que no se active hasta que el

monitor serial reciba información. Esto frena el sistema y evita las infinitas repeticiones de este. Para así no molestase más el LCD.clear(no lo puse en este código por problemas de espacio, pero estaría ubicado en la primera línea de este) borrando el texto junto con los LCD.print que lo vuelve a imprimir(lo que generaba que no se pudiese apreciar bien el texto mostrado por la pantalla).

| LCD.print("1 - Boton");  LCD.setCursor(0,1);  LCD.print("2 - Texto");  while(!Serial.available){  }  String opcion = lectura();  if(opcion == "1"){  }  else if(opcion == "2"){  } |
| --- |

Aunque el while no tenga nada dentro, lo que hace es que el código entre y se quede bloqueado dentro de este automáticamente hasta que reciba un valor por el monitor serial (el “!” refiere a negar un valor o a “False”, por lo que solo va a entrar en el while si la función Serial.available es negativa que será su valor hasta que se ingrese algo por el monitor serial).

Ahora que ya solucione este problema, voy a empezar por lo que ya tengo hecho. Así que ahora voy a hacer que funcione la opción 2:

como hay dos tipos de traducción de texto (Texto a Morse y Morse a Texto), tengo pensado hacer una subopción dentro de la opción 2, que me de a elegir la función que quiera utilizar, en caso de ingresar la opción 1(dentro de la subopción), te llevará al traductor de Morse a texto, mientras que si el ingreso es “2”, te llevará al traductor de Texto a morse.

| else if(opcion == "2"){  LCD.clear();  LCD.print("1 - Morse a TXT");  LCD.setCursor(0,1);  LCD.print("2 - TXT a Morse");  LCD.setCursor(0,0);  while(!Serial.available()){  }  if(Serial.available()){  String opcion2 = lectura();  if(opcion2 == "1"){  }  else if(opcion2 == "2"){  } |
| --- |

Básicamente, esta parte del código funciona exactamente igual que la de arriba (también hay un else para corroborar si el ingreso es diferente a “1” o “2”, pero no lo mostré por acá por que ocupaba demasiado espacio).

Junté todo este código principal en una función llamada “Principal” y lo puse dentro del loop para que cada vez que se ingrese un valor erróneo o se termine de procesar una función, vuelva a empezar en este código.

Ahora falta crear las funciones para llamar dependiendo de la opción:

Empecé creando la que ya tenía preparada para copiar y pegar (osea la de conversión de Morse a

Texto) y la llamé MorseATXT. No realicé ningún cambio en esta, ya que antes funcionaba correctamente.

Luego para la función del uso del botón para enviar señales en código morse y así traducirlas

a nuestro alfabeto, empecé pegando el código que hice para la función que ya tengo creada y le estoy cambiando cosas.

Lo primero que hice fue agregar al principio de la función el ciclo while que había usado en el código de prueba que hice para el uso del botón como traductor morse. La idea que tengo en mente, es que, como tanto en el traductor de texto en código morse a nuestro alfabeto como en el traductor de pulsaciones en morse a nuestro alfabeto, necesito un string para detectar y escribir el texto. Lo único que cambia es la manera de ingresar.

| while(true){  if (valor > 0 && valor < 120000){  caracter = '.';  }else if(valor >= 120000 && valor < 80000){  caracter = '-';  }else if(valor >= 80000 && valor < 120000){  caracter = '/';  }else if(valor >= 120000){  break;  } } |
| --- |

Como siempre digo, quité partes del código por problemas de espacio. Probé el código y todo andaba bien. No hubo ningún error ni nada que me lo complicara salvo que por mas que lo intentara no podía ingresar guiones o barras. Rápidamente me di cuenta del error, y es que tenía valores erróneos en los tiempos de pulsación del botón. Una vez corregido quedó de la siguiente manera(de todas formas, los valores son temporales, ya que los cambiaré en base a las respuestas del arduino físico. Tampoco puse instrucciones dentro de los if por que ya está arriba):

if (valor > 0 && valor < 100000){  
 }else if(valor >= 100000 && valor < 200000){  
 }  
 else if(valor >= 200000 && valor < 300000){  
 }else if(valor >= 300000){   
 }

Ahora solo me queda resolver la función de traducción de texto a código morse. El funcionamiento de este debería ser que el usuario ingresa el texto y el sistema lo pasa todo a mayúsculas automáticamente(para que puedan coincidir las letras con la de los vectores). Luego recorrer la palabra y comparar cada posición con los vectores que ya tengo creados mediante el mismo algoritmo que vengo haciendo para las otras funciones pero a la inversa. Una vez comparada la letra con el vector, en vez de imprimirla, devuelve el string y con el uso de otro for, recorre el string nuevo y mediante el uso de if y else if detectar si la posición es un punto o un guión y en base a eso prender un LED por más o menos tiempo. Luego imprimir la letra(en nuestro alfabeto).

Empecé pegando el mismo código que use para el traductor de morse a texto, y de ahí iré haciendo las respectivas modificaciones. Primero eliminé los if relacionados con barras, como por ejemplo el que revisa si la posición actual del string es o no es una barra y en base a eso va formando la letra en morse(lo cual no es necesario en esta función ya que las letras ya están traducidas y tengo que hacer el procedimiento contrario).

Analizando mi objetivo, me encontré con un problema. Resulta que yo quiero que la función buscar, me devuelva tanto el valor booleano de encontrado(que representa si se encontró el caracter ingresado traducido en morse) así como un string que sea la traducción en morse del caracter ingresado(en caso de que exista en los vectores). El problema es que yo no sé hacer que una función devuelva dos tipos de dato diferentes en arduino, por lo que cree una nueva función llamada buscarMorse, que funciona igual que la función buscar, solo que esta no imprime nada y en vez de devolver un valor booleano, devuelve un string:

| String buscarMorse(String caracter, int hastaDonde, String vectorComparar[], String vectorImprimir[]) {  String morse = "";  for (int n = 0; n < hastaDonde; n++) {  if (caracter == vectorComparar[n]) {  morse = vectorImprimir[n];  break;  }  }  return morse; } |
| --- |

Ahora que añadí esa función, la llamé pasandole los parametros pero me tiraba error el arduino. Estuve un rato viéndolo e intentando comprender por qué lo tiraba, pero rápidamente leí en el error que mencionaba algo de String y char. En ese momento me di cuenta que a la función le estaba pasando por parámetros “texto[i]”, lo que refiere a un caracter en cierta posición de un String, o en otras palabras un caracter(char), es decir, estaba comparando un String con un char y por eso me tiraba error. Mi solución a ese problema fue crear una variable String llamada letraActual sin asignarle un valor, para así más adelante poder igualarla a la posición actual del texto y convertir el char en un String. Con esto logré tener una base firme del código con la que desarrollar más funciones de este y estar más cerca de lograr el objetivo final del proyecto (hay más código pero no lo pongo por que si no me excedo de la capacidad de imágenes/código por página):

| void TXTAMorse(){  while(!Serial.available()){  }  if (Serial.available()) {  LCD.clear();  contPos = 0;  String letraActual;  String texto = lectura();  int longTexto = texto.length();  for (int i = 0; i < longTexto; i++) {  letraActual = texto[i];  String morse = buscarMorse(letraActual, 26, letras, letrasMor); |
| --- |

Se me ocurrió que no era necesario verificar si se encontró el caracter con la variable encontrado, si no que en vez de eso podría verificarlo con el valor de la variable con el nombre de morse(si no tiene valor significa que no se encontró el caracter):

| else if (morse == "" && letraActual != "") {  LCD.print("?");  } |
| --- |

Una vez con la letra traducida y con su equivalente en morse guardado en una variable, falta leer la variable y prender un LED más o menos tiempo dependiendo del valor de la posición en la que se encuentre del String en morse equivalente a la letra ingresada(esto mediante un for).

Antes de hacer el ciclo for, necesito ponerle un límite a la variable contadora, por lo que cree otra variable entera llamada longMorse que cambiará su valor mediante un longMorse = morse.length() siempre antes de entrar al ciclo for de la letra en morse. Ahora si, creamos el for con una variable m tipo int que se repita mientras sea menor al valor de longMorse y que por

cada recorrido sume su valor en 1. Mediante if y else if, verificamos si la posición es un punto o un guión y prendemos el LED por más o menos tiempo:

| longMorse = morse.length(); for(int m = 0; m < longMorse; m++){  if (morse[m] == '.'){  digitalWrite(led, HIGH);  delay(250);  digitalWrite(led, LOW);  }  else if(morse[m] == '-'){  digitalWrite(led,HIGH);  delay(500);  digitalWrite(led,LOW);  }} |
| --- |

Con todo esto hecho, solo faltan 5 cosas por hacer:

1 - Conexión por bluetooth, esto lo haré el martes con ayuda del profe, ya que dijo que me iba a enseñar)(el martes es la última clase en la que podemos seguir con el proyecto, por eso lo terminé en mi casa, ya que no puedo depender de mi grupo).

2 - Conversión del texto ingresado a mayúsculas en la función de traducción de texto a código morse(Busqué como hacerlo, pero no encontré mucho. Por lo que entendí, hay una librería que viene con una función que permite hacer esto, pero esa librería solo está disponible para el arduino físico, por lo que seguiré buscando y de última me fijaré el martes en la escuela).

3 - Revisión exhaustiva del código(Para buscar partes en la que el funcionamiento no sea como lo deseado).

4 - Modelo 3D del proyecto(Mañana en la escuela hablaré con mi grupo y les diré si pueden hacer algo de este, ya que yo hice toda la parte del código. De no ser así, trataré de hacer algo el martes en la escuela).

5 - Presentación del proyecto(soy muy malo con powerpoint/canvas. Trataré de hacerlo junto a mi grupo, aunque es de mis menores preocupaciones).

25 de noviembre

Mi idea de hoy, es ver si puedo solucionar lo del texto en mayúsculas. Pero antes, voy a hacer una revisión del código y fijarme si todo funciona bien.

Por alguna razón, si ingreso un texto en morse, para traducirlo a nuestro alfabeto no se muestran todas las letras, y muchas se borran o se muestran mal. Estoy poniendo diversos Serial.print por el código para hallar el error.

Por lo que veo, encuentro todo correcto. Hasta le pregunté a una IA y me dijo que el código estaba bien y no debería de malfuncionar. El código lee bien cuando hay dos barras y cuando hay una sola, acopla y separa bien la secuencia de puntos y guiones que hacen referencia a las letras o número. Sin embargo, hay letras que no imprime. Seguiré buscando la solución.

Probé poniendo un print directamente luego de ingresar el texto por el monitor serial. Comprobando las dos secuencias de caracteres, me di cuenta que eran diferentes, por lo que puedo deducir que esto es debido a un error de lectura. Cuando vi esto, recordé que al principio, también tenía un error de lectura, en el que se me duplicaban los caracteres, la solución que encontré a esto fue agregar un delay de 10 milisegundos, así que siguiendo esa lógica, le baje el delay de 10 a 2 microsegundos, lo que arreglo mi problema e hizo que el texto ingresado se lea correctamente.

Aprendí un nuevo comando para transformar un string a mayúsculas. Investigando un poco más encontré el uso correcto del comando, no necesitaba uso de ninguna librería, si no que no lo estaba escribiendo bien:

| String.toUpperCase() |
| --- |

La razón por la que no me funcionaba bien el comando y me tiraba error, es que lo estaba tratando de poner dentro de un Serial.print, y cuando vi que no podía por que me tiraba error, intenté declarar una variable igualándola a la función. Lo que tenía que hacer es simplemente poner una línea de código como la que está arriba y automáticamente el String se cambiaría todo a mayúsculas. También existe una función que se escribe igual, solo que en vez de cambiar el String a mayúsculas los cambia a minúsculas. Su nombre es toLowerCase().

| void TXTAMorse(){  while(!Serial.available()){    }  if (Serial.available()) {  LCD.clear();  contPos = 0;  int longMorse = 0;  String letraActual;  String texto = lectura();  texto.toUpperCase(); |
| --- |

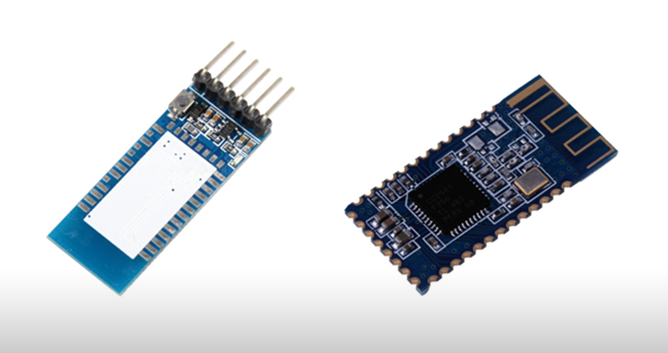
26 de noviembre

Hoy toca hacer la configuración por bluetooth y a verdad estoy medio nervioso ya que esta es

la última clase en la que podemos hacerle preguntas al profe (ya que la próxima se presenta todo). Entre mis compañeros de grupo (menos una), nos organizamos para que unos hagan el

modelo(que no lo hicieron todavía) y otros hagan la presentación.

Llamé al profe y luego de un rato en el que fue buscando un componente en las cajas de componentes de cada grupo, me trajo unos módulos extraños que supongo que son para el uso del bluetooth:



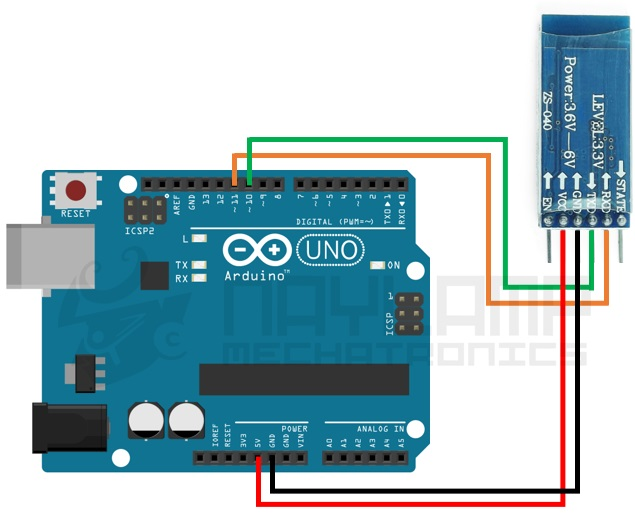
Me dijo que busque “Módulo bluetooth maestro y esclavo” en google. Aunque no me dijo que buscar respecto a esto, tras buscar un poco encontré un código que creo que me puede ser de

utilidad para la configuración de estos dos módulos, estoy esperando que el profe venga a ayudarme, así que cuando venga continuaré.

Ahora que vino el profe conmigo y tras buscar imágenes en google y algunas páginas encontramos las conexiones necesarias para el módulo:

Conexiones:

* STATE: No conectar.
* RXD: Cualquier pin digital.
* TXD: Cualquier pin digital.
* GND: GND del arduino(negativo).
* VCC: 5V(positivo).
* EN: No conectar.



Me acerqué a preguntarle al profe cómo diferenciar cada uno (ya que uno era el maestro y otro el esclavo). Me contó que por lo general el maestro era el módulo de tipo HC-05, mientras que el esclavo el módulo de tipo HC-06. Le pregunté cómo hacer para diferenciar cual era el HC-05 y cual el HC-06 y como él tampoco sabía, estuvo un rato investigando. Luego me dijo que parecía ser que ambos módulos servían de HC-05(osea que no había un módulo de tipo HC-06 entre estos), pero también me dijo que el módulo HC-05 puede servir tanto como de maestro así como de esclavo, por lo que pese a esto no debería haber tanto problema.

Para que no me volverme loco y no quemarme tanto la cabeza, el profe me dijo que trabaje con un solo arduino antes de pasar a trabajar con 2, así que una vez con los dos módulos ya conectados al arduino, empecé a probar copiando y pegando el código que encontré en una página:

<https://hetpro-store.com/TUTORIALES/configurar-bluetooth-hc-05-como-maestro-esclavo/>

| #include <SoftwareSerial.h> SoftwareSerial BTSerial(RxD, TxD); void setup(){ pinMode(KEY, OUTPUT); digitalWrite(KEY, HIGH); BTSerial.begin(38400); Serial.begin(9600);  }void loop(){ Serial.write(BTSerial.read()); BTSerial.write(Serial.read());} |
| --- |

Tuve que quitar algunas partes del código, ya que era demasiado largo. Traté de comprenderlo, añadiendo más define (hay unos define con sus respectivos pines que no puse por temas de espacio) y preguntando o investigando que hacía cada función. Finalmente, el profe me dijo que no me mate tratando de entender el código ni que le cambie nada (ni siquiera los pines), que solo copie y pegue que ya de por sí son difíciles de configurar los módulos bluetooth (como solo había código para un módulo, también me dijo que pruebe primero con uno y luego con otro individualmente y sin tenerlos conectados al mismo tiempo), pero seguía habiendo un problema,

y es que en el código en la parte de los define, a parte de haber uno para los pines RXD y TXD, había uno llamado KEY, el cual no estaba en los módulos que me había dado el profe, pero como no esta ahora conmigo le preguntaré luego.

Seguí investigando, y encontré otra página con un código diferente que se puede encontrar junto al paso a paso en la siguiente página:

<https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at.html>.

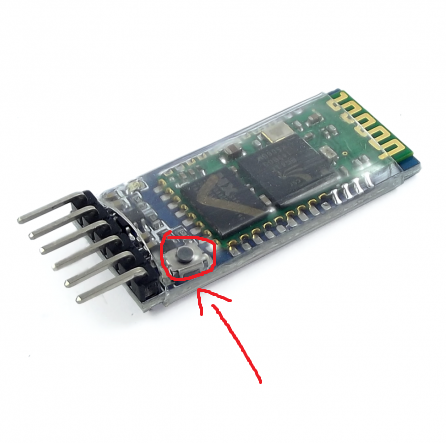
En la página cuentan que hay dos modos posibles para el módulo HC-05: AT 1 y AT 2. En el sitio mencionan que el modo que estarán utilizando será el AT 2, sin embargo si se utiliza el AT 1 tampoco debería haber ningún problema(nuevamente no está todo el código por problemas de espacio):

| SoftwareSerial BT(10,11);  void setup() {  BT.begin(9600);  Serial.begin(9600);  } void loop() {  if(BT.available())   {}  if(Serial.available())   {} } |
| --- |

¿Cómo acceder a los dos diferentes modos?

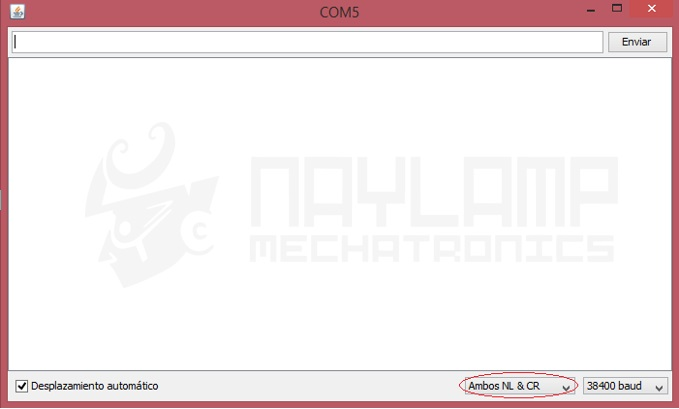
AT 1: Una vez conectado el módulo al arduino y el arduino conectado a la PC, tocar el botón que tiene en la parte trasera(el módulo). Una vez activado el modo AT 1, para asegurarse que funciona correctamente, el LED al costado del módulo debería empezar a prenderse y apagarse repetidamente y rápidamente.

AT 2: Utilizando el mismo botón que se utiliza para encenderlo en el modo AT 1, asegurarse de que el módulo esté desconectado(con que no esté recibiendo energía alcanza). Luego de tenerlo aislado de fuentes de energía, apretar y no soltar el botón mientras se conecta nuevamente el módulo(no hay que hacer los dos al mismo tiempo, primero apretás y mantenes apretado el botón y mientras lo mantenés apretado conectas el módulo). Al contrario que en el modo AT 1, si el procedimiento se realizó correctamente, el LED del costado del módulo deberá prenderse y apagarse repetidamente pero a una velocidad muchísimo más lenta.



Inicié el modo AT 2 y conecté los pines según me dice el código de la página que encontré. Inicié el monitor serial y según el sitio si escribía AT, debería de devolver OK aunque por alguna razón esto no sucedía, si yo ingreso AT no devuelve nada, pero como el profe está ocupado ahora me tocará esperar a que esté disponible para pedirle ayuda.

Leyendo un poco más detenidamente la página, vi que me faltaron hacer algunas configuraciones en el monitor serial:



Como lo dice esa imagen, tenía que cambiar esa ventanita a la opción que se ve en pantalla, así como también tenía que cambiar la velocidad por una de 38400 baud, ya que viene por defecto en 9600 baud. Una vez hecho esto cerré el monitor serial, desconecte y volví a conectar el arduino por las dudas, revisé las conexiones que no hubiese nada mal conectado, encendí el módulo HC-05 en modo AT 2, abrí el monitor serial y volví a mandar el comando AT. Pese a todo esto, el monitor serial por alguna razón, seguía sin devolver ningún valor, así que tendré que seguir esperando a que venga el profe a ayudarme.

Tras un rato de analizar cuál podría ser el problema junto al profe, este me sugirió que cambiara el numero dentro del Serial.begin por 38400 para quedar acorde a la velocidad del monitor serial que tuve que cambiar por la página. Una vez hecho esto, volví a probar pero seguía sin funcionar, hasta que en un momento tras conectar y desconectar varias veces, se me olvido ponerlo en el modo AT 2, y active sin darme cuenta el modo AT 1 y tras probar una vez más ingresando AT, el monitor serial devolvió OK.

No entiendo por qué surgió este problema ni por qué no funciona con el modo AT 2, de todas formas seguí probando los comandos y encontré otro error más. Pese a que con la configuración que cambié y estar usando el modo AT 1, al ingresar cualquier tipo de comando del bluetooth exceptuando AT, me devuelve error 0, por ejemplo si yo ingreso AT+NAME=Robot (que debería devolver OK) que es un código para cambiar el nombre del módulo, en vez de decir OK devuelve error 0. Seguiré investigando y veré si lo puedo arreglar.

Junto al profe, seguimos probando alterando un poco el código, o desconectando y volviendo a conectar tanto el arduino como el módulo bluetooth con sus dos modos. Como el profe me había dado dos módulos, probé con el otro, con el cual resultó de la misma manera(el modo AT 2 no funcionaba, y tenía que usar el AT 1. En el modo AT 1 solo funcionaba el comando AT a

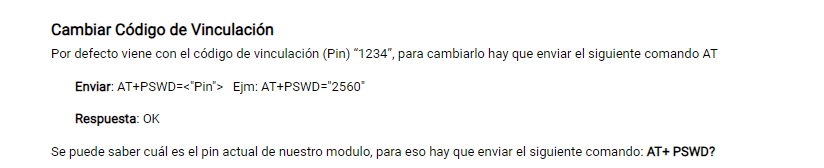
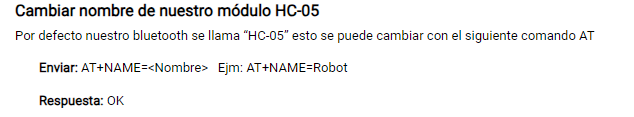
solas, ya que si ingresaba cualquier otro comando con AT y algo mas daba error).

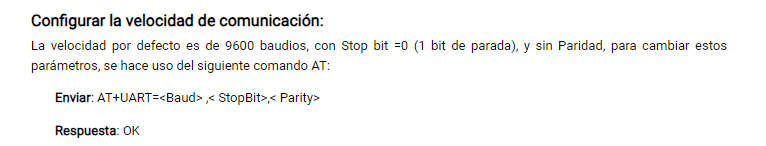
Finalmente, trás mucho rato de prueba y error, el profe fue a buscar otro módulo (creo que también HC-05, pero no estoy seguro ya que no lo comprobé) y me lo dio para que probara con ese. En lo que lo conectaba, un chico de 5to estaba en el aula hablando con los compañeros que tenía al lado, y vio que estaba trabajando con los módulos bluetooth HC-05 y mencionó que el año pasado él también trabajó con ese tipo de módulo y que aunque el profe le había dado varios ninguno funcionaba. Trás escuchar eso, me dispuse a hacer un último intento con el módulo que recién me había traído el profe, y en caso de no funcionar no usaría el bluetooth como parte del resultado final del proyecto.

Empecé conectando el módulo de la misma manera que había conectado los otros, aunque al momento de prenderlo dudé por un momento si lo había conectado bien por que no veía el parpadeo del LED rojo al costado del módulo, pero tras un breve análisis observé que si tenía LED pero era de color verde(similar al color del módulo y por eso no lo ví). Empecé probando el modo AT 2 y nuevamente no funcionaban los comandos AT, también establecí el valor del Serial.begin nuevamente a 9600 para probar si cambiaba algo pero siguió sucediendo lo mismo. Inicié el módulo HC-05 en el modo AT 1 y probé ingresando nuevamente los comandos del AT, pero esta vez no devolvía nada. Ahora estoy viendo por que ahora no devuelve ni siquiera error 0 u OK.

Encontré el error, olvidé volver a cambiar el valor entre los parámetros del Serial.begin de 9600 a 38400. Solucionado esto, el código volvió a funcionar como antes, es decir, si yo ingreso AT, la consola devuelve OK, pero si ingreso por ejemplo AT+NAME=BT1, devuelve error 0(tanto con ese comando como con cualquiera de los comandos de las siguientes imágenes,

exceptuando AT solo):



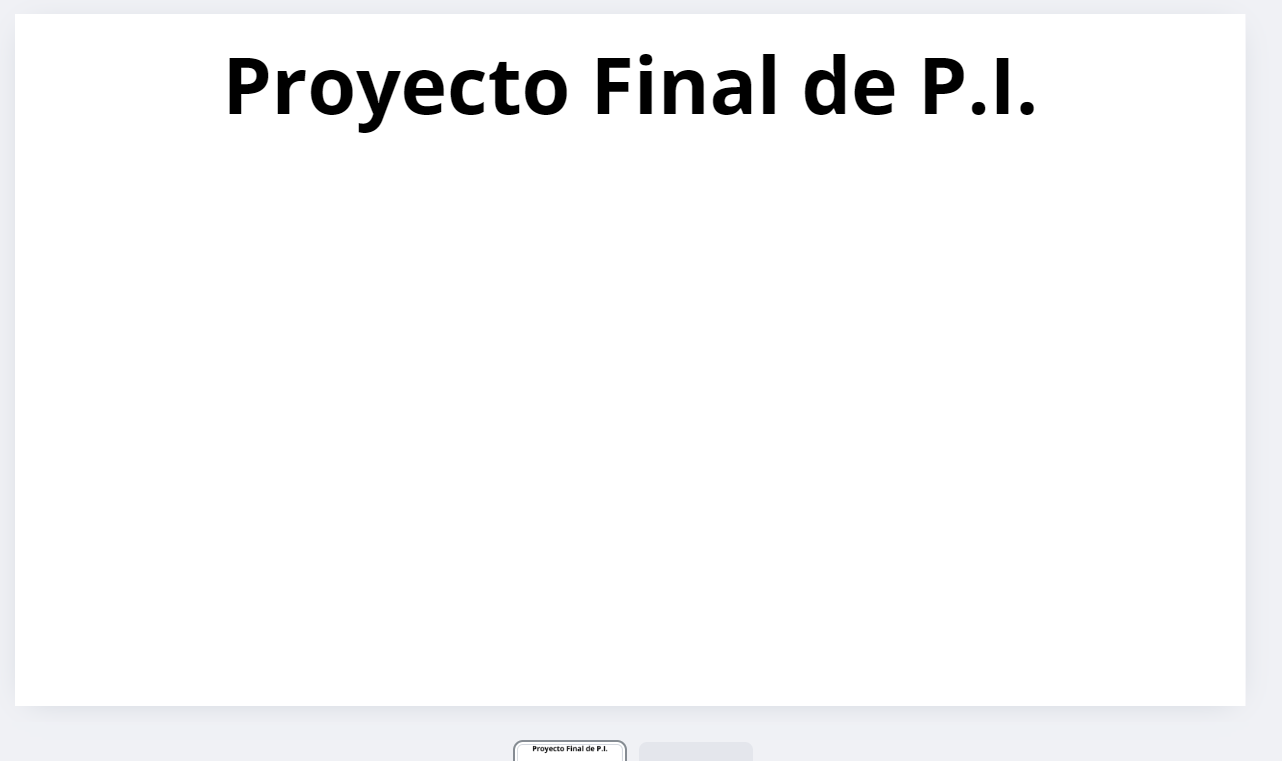


Hay bastantes más comandos, pero por temas de espacio no los puedo poner todos. Pero para hacernos una idea estos permiten configurar casi todo tipo de aspectos en cuanto al módulo de bluetooth HC-05 se refiere.

Volviendo al problema principal, no encontramos solución ni el por que el error junto al profe, por lo que finalmente quedará descartada la comunicación inalámbrica entre dos arduinos por medio de bluetooth y los módulos HC-05. La verdad, me pareció interesante, aunque bastante complicado de hacer (también por que el profe dijo varias veces que son muy difíciles y mucho lío de configurar). Una pena que finalmente no se pudiera implementar el bluetooth para el resultado final del proyecto, aunque el lado bueno es que es menos trabajo para mi y para el grupo en general.

Hablando del grupo, logré que hagan algo (o por lo menos eso parece):

ahora mismo, hay una persona haciendo el modelo 3D del proyecto (el cual va a tener que modificar por que descartamos el uso del bluetooth y por ende usaremos un solo arduino)(pondría una imagen, pero no tengo una imagen de como va, aunque si lo vi presencialmente y va tomando forma). Por otro lado, el resto parece estar haciendo la presentación, aunque esto en realidad es una media verdad ya que en todo el día de hoy hicieron tan solo la carátula(de hecho ni siquiera eso, solo pusieron el título del proyecto y no pusieron nada más):

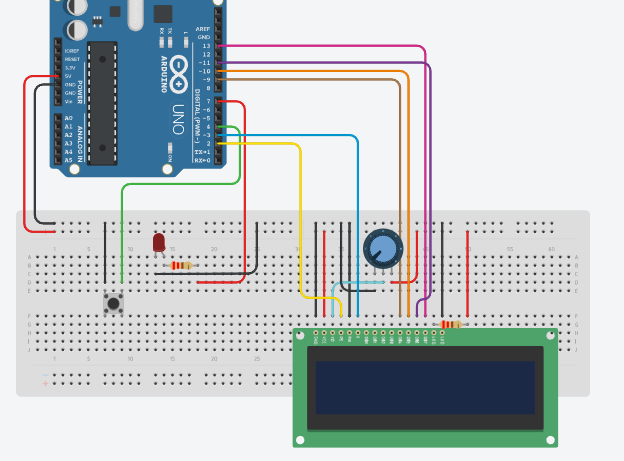


Por lo que veo, tendré que intervenir en la realización de la presentación a no ser que se pongan las pilas y empiecen a hacer cosas. De todas formas, tenía pensado hacerlo ya que necesitan información sobre los errores que se presentaron y cosas relacionadas y como fui el único que hizo algo, si yo no intervengo no tendrían nada que escribir respecto a estos.

Por otro lado, ayer el profe subió al classroom la tan misteriosa “Nota X”. No la había abierto hasta ahora porque cuando la subió yo me metí y era un formulario que tenía que solicitar acceso. Hoy en clase algunos compañeros mandaron su formulario y pude ver lo que era, aparte de que ya no tenía que solicitar acceso, por lo que abrí el formulario, seleccioné el proyecto de mi grupo y califiqué a mis compañeros en base a lo que trabajaron en el proyecto con las notas disponibles (del 1 al 4). Un rato antes de que mande el formulario, una persona de mi grupo se me acercó a decirme cosas como de que había trabajado, que tenía cosas en la carpeta de campo, que estuvo investigando sobre el proyecto, y más cosas (justo la persona que menos hizo de mi grupo por no decir que no hizo nada, y eso que nadie hizo casi nada excepto yo). Obviamente me di cuenta que me lo decía para que no le ponga la peor nota en la nota x, y que muy probablemente no había hecho nada en su carpeta de campo. Me insistió un rato que si quería ver lo que llevaba hecho, pero no le vi lo necesario, así que le dije que no y luego complete el formulario.

Ahora sí, con nada más que decir ni que detallar. Procederé a explicar el funcionamiento del código y de cada una de sus funciones:

Conexiones(en tinkercad):



La idea del proyecto es que al iniciar el código te aparezca un menú con 2 opciones en la pantalla LCD donde para elegir una opción el usuario deberá ingresar datos por la consola serial mediante el uso del teclado. Las opciones serían 1 - Botón y 2 - Texto. Si el usuario ingresa cualquier otro número diferente a 1 o 2(aunque son tratados como String) o si ingresa cualquier cosa que no sea ni 1 ni 2, se mostrará un mensaje por la consola serial diciendo que el ingreso es incorrecto y redirigirá el código directamente al menú principal. Esta función la llamé principal y está ubicada en el loop:

| void principal(){  LCD.clear();  LCD.print("1 - Boton");  LCD.setCursor(0,1);  LCD.print("2 - Texto");  while(!Serial.available()){  }if (Serial.available()) {  String opcion = lectura();  if(opcion == "1"){  }else if(opcion == "2"){  }  } } |
| --- |

Cabe aclarar que el código está considerablemente recortado para que entre en la página y no se exceda del tamaño permitido.

En caso de que el ingreso sea “1”, el código se comportará de la siguiente manera:

El usuario deberá presionar el botón por más o menos tiempo. Dependiendo del tiempo que lo mantenga presionado, el sistema guardará un valor u otro (punto guión o barra). En caso que se mantenga apretado por mucho tiempo, se cortará el ciclo del botón y se juntarán todos los

caracteres generados mediante las pulsaciones del botón para convertirse en un String, el cual luego se procesará y se convertirá en una letra o número. A esta función la llamé BotonATXT, pero también utilicé otras funciones que cree llamadas buscar y lectura:

La función de buscar utiliza sus parámetros para igualar un string con un vector que contiene todas las traducciones de números y letras en morse varias veces y con todas sus posiciones mediante un ciclo for. Si en algún momento del ciclo for encuentra que el String es igual a una posición del vector, imprime la misma posición del vector pero en un vector que contiene todas las letras y números traducidas a nuestro alfabeto, y también cambia la variable encontrado a true (esto es para a futuro, verificar si existe el string ingresado y si imprimir la letra u otra cosa).

| bool buscar(String string, int hastaDonde, String vectorComparar[], String vectorImprimir[]) {  bool encontrado = false;  for (int n = 0; n < hastaDonde; n++) {  if (string == vectorComparar[n]) {  LCD.print(vectorImprimir[n]);  encontrado = true;  break;  }  }  return encontrado; } |
| --- |

La parte importante de la función BotonATXT, es la que estará a continuación. Esta consiste en un ciclo while que se repite constantemente, con excepción de un break que solo se ejecutará si el botón se mantiene presionado cierto tiempo, de lo contrario, el sistema interpretará cuanto tiempo se mantuvo apretado el botón mediante el uso de la función pulseIn(que detecta cuánto tiempo tarda una variable a volver a su valor original desde que se alteró su valor ) y empezará a juntar en un string un carácter u otro dependiendo del tiempo que se mantuvo apretado el botón (si es muy poco será un punto, si es un valor intermedio será una línea, si es mucho será una barra). En caso que no se presione el botón por mucho tiempo (osea que el valor que devuelva el pulseIn sea igual a 0), se mostrará un mensaje por el monitor serial mencionando que no se registró ningún valor, aunque no se cortará el código, si no que seguirá esperando a detectar un valor saliente del pulseIn(o sea de la pulsación del botón), es decir que la única manera de frenar el ciclo while sería manteniendo apretado el botón por un tiempo mayor a 200000 microsegundos para que el código entre por el tercer else if ejecutar el break:

| void BotonATXT(){  char caracter;  while(true){  caracter;  float valor = pulseIn(boton, LOW, 3000000);  if (valor > 0 && valor < 100000){  caracter = '.';  }else if(valor >= 100000 && valor < 150000){  caracter = '-';  }else if(valor >= 150000 && valor < 200000){  caracter = '/';  }else if(valor >= 200000){  break;  }  } |
| --- |

Como siempre, tuve que dejar solo lo esencial del código (y menos aún). A parte de lo que se vé, luego de que el sistema detecte un caracter u otro, prende un LED por más o menos tiempo. Luego de tener el String, el código entra en un ciclo for donde compara los Strings y los imprime, pero no voy a entrar en detalle de eso.

En caso de ingresar la opción 2, lectura esperará a que se ingresen datos por la consola serial. Una vez detecta que la consola está siendo utilizada(osea que se ingresaron datos mediante un Serial.read) esta función entra en un ciclo while para juntar los caracteres ingresados en un string mediante el uso de un ciclo for(va añadiendo caracter por caracter al String por cada recorrido) (por que aunque ingreses un String el monitor serial solo lee como char y creo, aunque no estoy seguro, que también int). Una vez juntados los caracteres devuelve la variable con el String(el delay tan pequeño, es para evitar errores ya que a veces el ciclo va tan rápido que se repiten caracteres o no se llegan a registrar algunos):

| String lectura() {  String a = "";  while (Serial.available()) {   char caracter = Serial.read();  a += caracter;  delay(2);  }  Serial.println(a);  return a; } |
| --- |

Acto seguido, se volverán a mostrar otras dos opciones (provenientes de la función principal)

dando a elegir el modo de traducción (Morse a texto o Texto a morse). Luego esperará a que el usuario ingrese algo por el monitor serial para repetir el mismo algoritmo, solo que esta vez, en caso de ingresar “1” te llevará a la función de traducción de código morse a texto, mientras que si lo ingresado es “2” te llevará a la función de traducción de texto a código morse(recortado por problemas de espacio):

| else if(opcion == "2"){  LCD.print("1 - Morse a TXT");  LCD.setCursor(0,1);  LCD.print("2 - TXT a Morse");  while(!Serial.available()){  }  String opcion2 = lectura();  if(opcion2 == "1"){  MorseATXT();  }  else if(opcion2 == "2"){  TXTAMorse();  }  else{  Serial.println("No existe esa opcion.");  }} |
| --- |

Si elegimos la subopción 1, se llamará a la función MorseATXT. Esta será la encargada de realizar la traducción de un string en morse a un texto/palabra/letra o número en nuestro alfabeto, y funciona de la siguiente manera:

Una vez empieza la función, gracias a la función lectura, se repite el mismo algoritmo que viene realizando el código anteriormente. Ahora sí, cuando la función obtiene el String, entra en un ciclo for en el que va comparando cada posición del String con el caracter ‘/’, si es diferente, añade el caracter a una variable tipo String llamada “caracter”, que, como su nombre lo indica representará al caracter (en nuestro alfabeto) que significan el conjunto de puntos o guiones en morse hasta que haya una barra. Luego de esto, gracias a la función buscar, compara al String caracter con los vectores de letras y números ya armados utilizando las traducciones que saqué de la imagen que está al principio de todo, y si el String es igual, imprime la misma posición pero del vector de letras o números normales. Como la función buscar devuelve una variable de tipo booleano (encontrado), usé este para verificar si se encontró la equivalencia de caracter en el vector de letras o números en morse y así poder realizar ciertas acciones como mostrar un signo de interrogación (‘?’) en caso de no ser encontrado.

Por último, MorseATXT tiene la capacidad de, en base a la posición de una variable contadora(contPos), la cual se suma cada vez que se escribe un caracter o un espacio en la pantalla LCD, poner el cursor en una posición u otra, así como en caso de que el texto exceda la capacidad de la pantalla (32), mostrar el texto escrito por 3 segundos, borrar la pantalla y mostrar lo siguiente. Finalmente, una vez mostrado el mensaje, se mostrará un texto enviado por el monitor serial indicando que todo el mensaje ha sido escrito y que el contenido de la pantalla se borrará a a los 5 segundos. Luego de esto, el código volverá a empezar y esperará a recibir información por el monitor serial:

| void MorseATXT(){  while(!Serial.available()){  }  LCD.clear();  String texto = lectura();  for (int i = 0; i < longTexto; i++) {  if (texto[i] != '/') {  caracter += texto[i];  }  if (texto[i] == '/' || i + 1 == longTexto) {  if (contPos < 16) {  LCD.setCursor(contPos, 0);  } else {  LCD.setCursor(contPos - 16, 1);  }  if (!encontrado) {  encontrado = buscar(caracter, 10, numerosMor, numeros);  }  if (!encontrado && caracter != "") {  LCD.print("?");  }  contPos++;  if (contPos > 31) {  Serial.println("El texto es demasiado largo.”);  contPos = 0;  }}}  Serial.println("Mostrando el texto.”);  delay(5000); } |
| --- |

Como última función, en caso de ingresar como subopción “2”, la función principal te llevará a la función TXTaMorse. En esta función el usuario deberá ingresar un mensaje mediante el uso de su teclado(sobra decir que en nuestro alfabeto y no en el morse) y el mensaje ingresado se mostrará por la pantalla LCD(tal cual fue ingresado) caracter por caracter mientras a su vez se prende y apaga por más o menos tiempo dependiendo de los caracteres equivalentes a la letra o número en morse repetidas veces un LED(es decir, se escribe un caracter y hasta que no termine de hacer su secuencia de puntos y líneas el LED no se imprimirá el siguiente). Funciona de la siguiente manera:

Como en MorseATXT, esta función utiliza la función lectura para obtener un String para luego compararlo con diferentes vectores de Strings y así obtener su equivalente en el otro alfabeto. La diferencia entre estos dos, es que mientras MorseATXT utiliza la función buscar para realizar este procedimiento, TXTAMorse utiliza buscarMorse(que su código está más arriba), la cual funciona bastante parecida pero con la diferencia de que no hace el print de su equivalencia en la pantalla LCD y devuelve un tipo de dato diferente(esto lo hice porque necesitaba el String de la traducción de la letra o número en morse para determinar el tiempo que debía permanecer encendido el LED).

Cuando la función guarda el String, en vez de hacer la comparación de la posición actual con el caracter ‘/’ y en base a eso añadir el caracter a un string o realizar ciertas acciones, simplemente iguala la posición a un String(convirtiéndolo de char a String), ya que de otra manera, no lo podría usar como parámetro para la función buscarMorse(ya que esta pide String y no lo puedo cambiar a que pida un char por que tiene que igualar su valor al de otro vector en donde todas las posiciones son Strings).

Una vez obtenido el string equivalente a la traducción en morse de la letra o número en la posición actual del String ingresado(si es que no existe muestra por la pantalla LCD un signo de interrogación), imprime la posición actual del String por la pantalla LCD y le asigna un valor a una variable tipo int con valor 0 llamada longMorse con la función morse.length()(morse es el nombre de la variable tipo String que contiene la traducción en morse del caracter), que como su nombre lo indica calcula el largo del string de la letra o número en morse. Luego entra en un ciclo for en el cual recorre todo el String obtenido gracias a la función buscarMorse y por cada punto prende el LED por un tiempo, mientras que si el caracter en la posición actual es una línea lo prenderá por un tiempo diferente, aunque eso sí, el tiempo que se mantiene apagado es indiferente sin importar el tipo de caracter. Finalmente también, utiliza el mismo algoritmo utilizado en las otras dos funciones(BotonATXT y MorseATXT), es decir, hace la revisión de la posición siempre antes de imprimir cada letra y en base a eso pone el cursor de la pantalla LCD en una posición u otra, además que también si el contador se excede del límite de la pantalla, el texto faltante se mostrará pasados tres segundos. Una vez terminado el código el mensaje se mostrará por 5 segundos junto a un mensaje en el monitor serial avisando de esto, acto seguido

se borrará la pantalla y volverá a empezar el código:

| void TXTAMorse(){  while(!Serial.available()){  }  contPos = 0;  int longMorse = 0;  String letraActual;  String texto = lectura();  texto.toUpperCase();  int longTexto = texto.length();  for (int i = 0; i < longTexto; i++) {  if (contPos < 16) {  LCD.setCursor(contPos, 0);  } else {  LCD.setCursor(contPos - 16, 1);  } |
| --- |

Como el código es muy largo, lo iré explicando por partes. Para empezar, acá definimos la función creamos las variables que utilizaremos a lo largo del código: estas son contPos, que será una variable contadora(le sumaremos 1 cada vez que se imprima un caracter por la pantalla LCD), y también nos servirá para ubicarnos en que posición de la pantalla LCD estamos para así realizar diversas operaciones; longMorse, que nos permitirá(a futuro) saber la longitud de un String en específico para poder ponerle un límite a un for; letraActual, que será utilizado para convertir la posición actual de la variable texto en un String; y texto, que es el conjunto de caracteres(String) ingresados por el monitor serial. Aparte de todas estas variables, también hay una línea de código(texto.toUpperCase()) que lo que hace es convertir el String texto a mayúsculas, para así facilitar la comparación de vectores.

También, como podemos ver a principios del ciclo for, hay unos if que nos permiten verificar la posición del cursor en la pantalla LCD y dependiendo del valor de este, se coloca el cursor en una posición de la pantalla LCD o en otra. Esto se logra obtener mediante la corroboración de si

se encuentra en la mitad de arriba(los primeros 16 espacios) o en la de abajo (del espacio 17 al 32).

| letraActual = texto[i];  String morse = buscarMorse(letraActual, 26, letras, letrasMor);  if (morse == "") {  morse = buscarMorse(letraActual, 10, numeros, numerosMor);  }  if(morse == "" && letraActual == " "){  LCD.print(' ');   }  else if (morse == "" && letraActual != "") {  LCD.print("?");  }  if(morse != "" && letraActual != ""){  LCD.print(letraActual);  } |
| --- |

Esta porción de código, como ya explique anteriormente convirtiendo la posición actual del String texto en base al número de ciclos realizados por el ciclo for para luego ester poder ser suministrado como un parámetro en la función buscarMorse y así instanciar el String morse, el cual en lo que resta de código será utilizado como verificador de sí se encontró el equivalente del texto en morse o no, aparte de también ser utilizado(en otra parte del código) para mostrar la traducción en morse por un LED. Podemos ver que en la tercera línea de este fragmento de código verifica si la variable morse tiene algún valor guardado, y en caso de que no ejecuta nuevamente la función buscarMorse pero con otros parametros (esto es para que busque primero en el vector de letras y luego en el vector de números). Si no encuentra un equivalente a la letra actual, pero detecta que la letra actual es un espacio entonces mostrará un espacio por la pantalla LCD, y por último hace una última verificación de si tanto la variable morse como la variable letraActual tienen algún valor guardado para así poder imprimir por la pantalla LCD la letraActual del string ingresado al principio del código por el monitor serial.

| longMorse = morse.length();  for(int m = 0; m < longMorse; m++){  if (morse[m] == '.'){  digitalWrite(led, HIGH);  delay(250);  digitalWrite(led, LOW);  }  else if(morse[m] == '-'){  digitalWrite(led,HIGH);  delay(500);  digitalWrite(led,LOW);  }}   contPos++; } |
| --- |

Esta es la última parte del código, empieza asignando un valor a longMorse(ya instanciada antes) para instantáneamente iniciar un ciclo for utilizando esta variable como límite. El objetivo de este ciclo es recorrer cada posición del String morse y en base al caracter en cada posición prender el led por un tiempo reducido o por un tiempo mayor. Finalmente, cierra el ciclo for y aumenta en uno el contador de posiciones para continuar con el mismo algoritmo.

Y así finaliza mi carpeta de campo, la verdad a veces fue frustrante hacerla y otras veces me sentía desganado, así como también hubo días en los que me sirvió para relajarme y despejar la mente. Me llevo una experiencia nueva al haber hecho un proyecto como este principalmente solo y en un tiempo reducido(por lo menos desde mi perspectiva lo fue). Ahora solo falta la presentación, la cual espero que todo salga bien y aprobar la materia y por ende el proyecto por el que tanto me esforcé.