SISTEMA DE PROCESAMIENTO DE DATOS

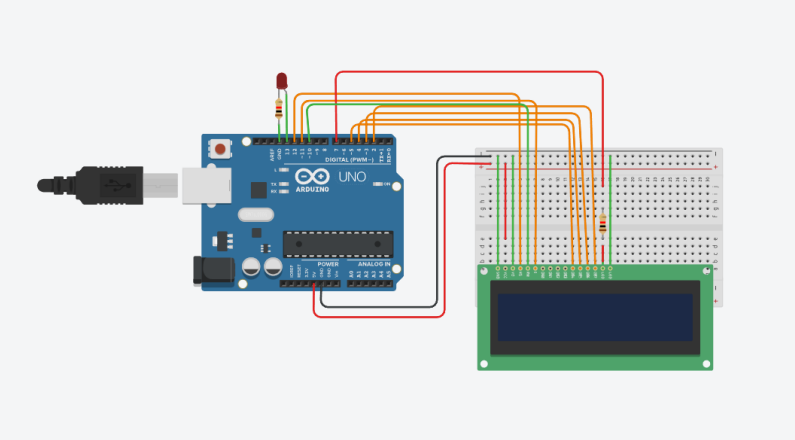
PROGRAMACIÓN DE AROMATIZADOR

Introducción

El trabajo práctico consiste en la programación de una placa Arduino para así controlar el intervalo de disparo de un aromatizador automático. El intervalo debe poder configurarse en un rango de entre 1 y 60 minutos a través de la computadora y del teclado del display; también debe poder colocarse la hora (HORA-MINUTOS) en el LCD; además cada vez que el aromatizador dispare debe no solo hacerlo sino que debe aparecer un dibujo en el LCD que desaparezca luego de un momento; se debe poder activar y desactivar la ejecución de los disparos cada vez que se solicite escribiendo estas misma palabras.

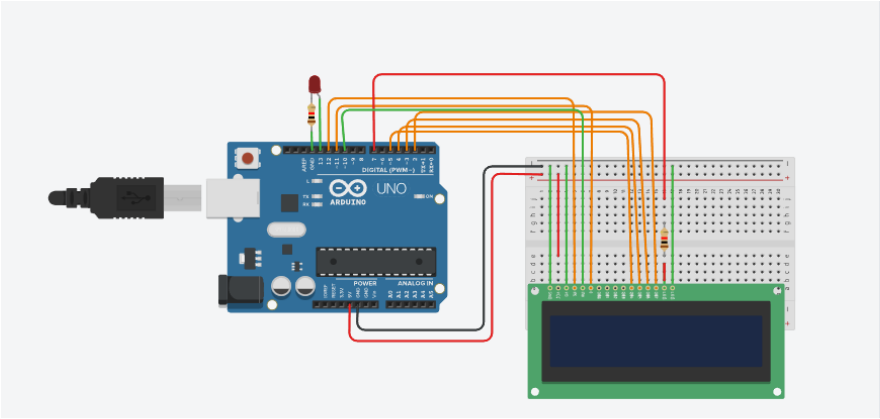
CONEXIÓN

Comenzamos conectando el LCD a la placa Arduino en el Tinkercad para hacer pruebas; tuvimos algunas dificultades ya que cuando lo conectamos al principio la pantalla no funcionaba, pero luego de corregir algunas conexiones y hacer otras que nos había faltado conectar, logramos que funcione correctamente. Optamos por poner un led para hacer las pruebas de la señal emitida por la placa ya que nos pareció la forma más simple de hacer las mismas.



PROGRAMACIÓN

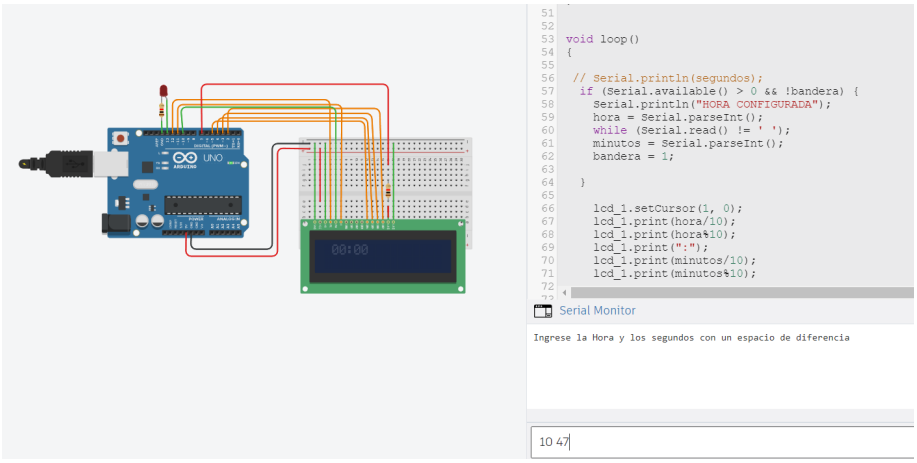
Comenzamos programando la configuración de los intervalos de disparo y el reloj. En primera instancia lo hicimos utilizando el comando “delay(1000)” para así contar un segundo cada vez que se ejecute, pero luego se nos indico que el tiempo debía medirse a través del comando “millis()” ya que así el programa nunca se pausa en el delay sino que siempre se está ejecutando. Para esto cambiamos la forma de calcular los segundos utilizando una estructura “ if “ la cual pregunta si millis() entre 1000 equivale a un contador “ x “ el cual comienza en 0 y se incrementa en 1 cada vez que la estructura se ejecuta. Otro problema que tuvimos al cambiar el delay por millis, fue que el dibujo en el LCD pasaba tan rápido que no llegaba a aparecer en la pantalla ya que antes el delay estaba entre el comando que hacía aparecer el dibujo y el que lo hacía borrar, por lo que el programa esperaba hasta que pase 1 segundo antes de hacer esto último. Lo solucionamos creando una estructura “ if “ la cual contiene los comandos para borrar el dibujo y para apagar la señal en el led; la misma se ejecuta cuando un contador dentro de la estructura “ if “ que calcula los segundos, llega a 1.



```
103
104     }
105
106     if((millis()/1000)>=x)
107     {
108         segundos++;
109         contSegundo++;
110         x++;
111     }
112
113
114     if(contSegundo==1)
115     {
116
117         digitalWrite(pin,LOW);
118         lcd_1.setCursor(8,0);
119         lcd_1.print(" ");
120
121     }
122
123
```

Reloj

La configuración del reloj optamos por hacerla utilizando una estructura que preguntara si había algo disponible en el buffer y si la bandera ,que habíamos declarado e inicializado primeramente, estaba en false. Dentro de esta estructura se lee la hora hasta que se introduzca un espacio para luego leer los minutos.



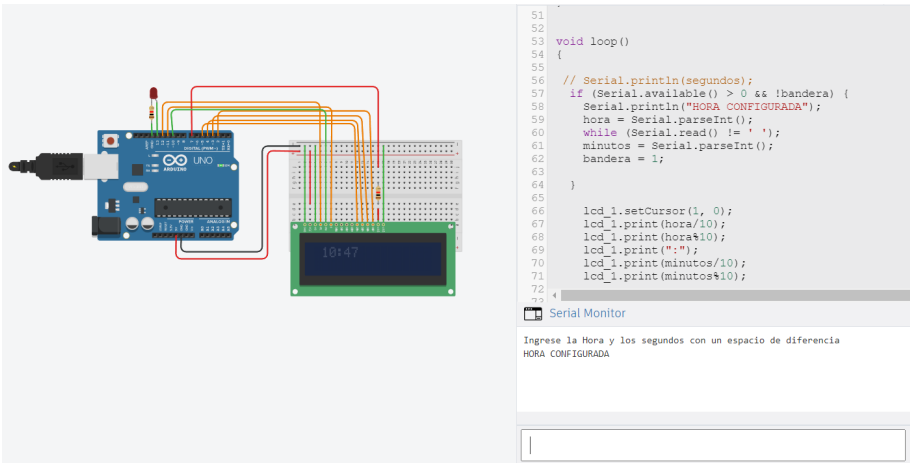
```
51
52
53  void loop()
54  {
55
56   // Serial.println(segundos);
57    if (Serial.available() > 0 && !bandera) {
58      Serial.println("HORA CONFIGURADA");
59      hora = Serial.parseInt();
60      while (Serial.read() != ' ');
61      minutos = Serial.parseInt();
62      bandera = 1;
63
64    }
65
66      lcd_1.setCursor(1, 0);
67      lcd_1.print(hora/10);
68      lcd_1.print(hora%10);
69      lcd_1.print(":");
70      lcd_1.print(minutos/10);
71      lcd_1.print(minutos%10);
72
```

Serial Monitor

Ingrese la Hora y los segundos con un espacio de diferencia

10 47

Ejecutados los comandos, aparece un mensaje de confirmación en el monitor Serial y la hora indicada se muestra en la pantalla LCD separando horas y minutos con dos puntos.




```
void loop()
{

 // Serial.println(segundos);
  if (Serial.available() > 0 && !bandera) {
    Serial.println("HORA CONFIGURADA");
    hora = Serial.parseInt();
    while (Serial.read() != ' ');
    minutos = Serial.parseInt();
    bandera = 1;

  }

    lcd_1.setCursor(1, 0);
    lcd_1.print(hora/10);
    lcd_1.print(hora%10);
    lcd_1.print(":");
    lcd_1.print(minutos/10);
    lcd_1.print(minutos%10);
```

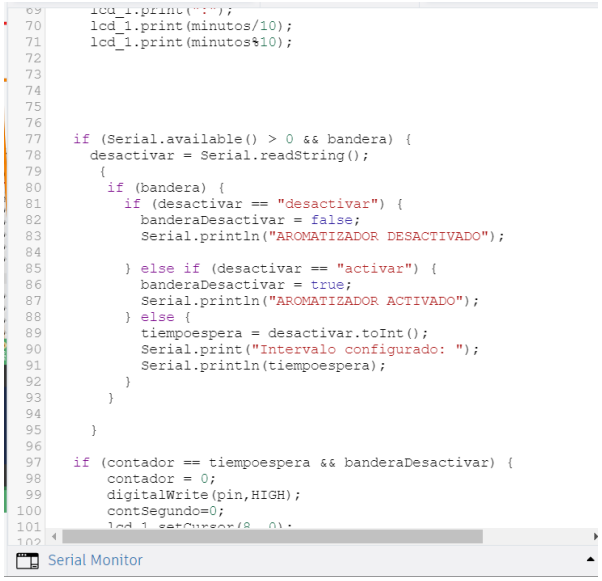
La mayor dificultad del trabajo fue hacer que al introducir la palabra desactivar o activar, se desactive o active el aromatizador. En primera instancia intentamos hacerlo creando una estructura " if " independiente al resto que preguntara si había algo en el buffer y luego dentro de esta si la palabra era desactivar la bandera se pone en false y si la palabra era activar, la bandera se ponía en true.

```
void setup()
{
  pinMode(A0, INPUT);
  servoutn.attach(3, 544, 2500);

   serial.begin(9600);
}

void loop()
{
     (Serial.available() > 0 && bandera) {
    if (Serial.length();
    tiempoespera = Serial.parseInt();
    Serial.println("INTERVALO CONFIGURADO");

  }

 if (Serial.available() > 3 && bandera)
 {
    desactivar = Serial.readString();
   if (desactivar == "desactivar")
   {
     banderaDesactivar = false;
     Serial.println("AROMATIZADOR DESACTIVADO");

   }
   else if (desactivar == "activar")
   {
     banderaDesactivar = true;
     Serial.println("AROMATIZADOR ACTIVADO");
   }
 }
}
```

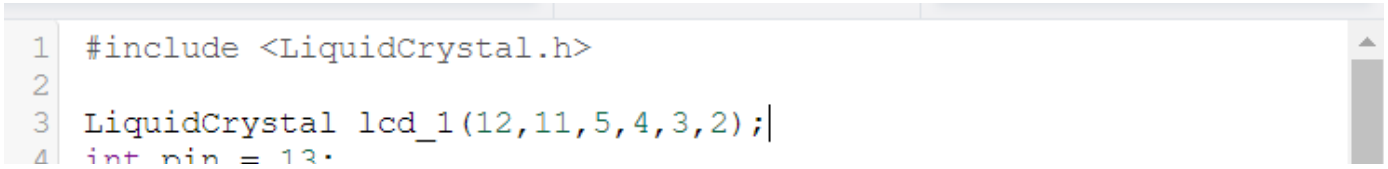
Serial Monitor

Esto no funcionaba ya que dependiendo en qué momento introduces la palabra, lo lee en el primer if o en el segundo. Esto se debe a que el comando Serial.available() lee si hay algo

en el buffer pero no devuelve cuantos caracteres hay. Este problema lo solucionamos creando una única estructura if en la cual dependiendo del dato ingresado, lo activa, lo desactiva o configura el intervalo.



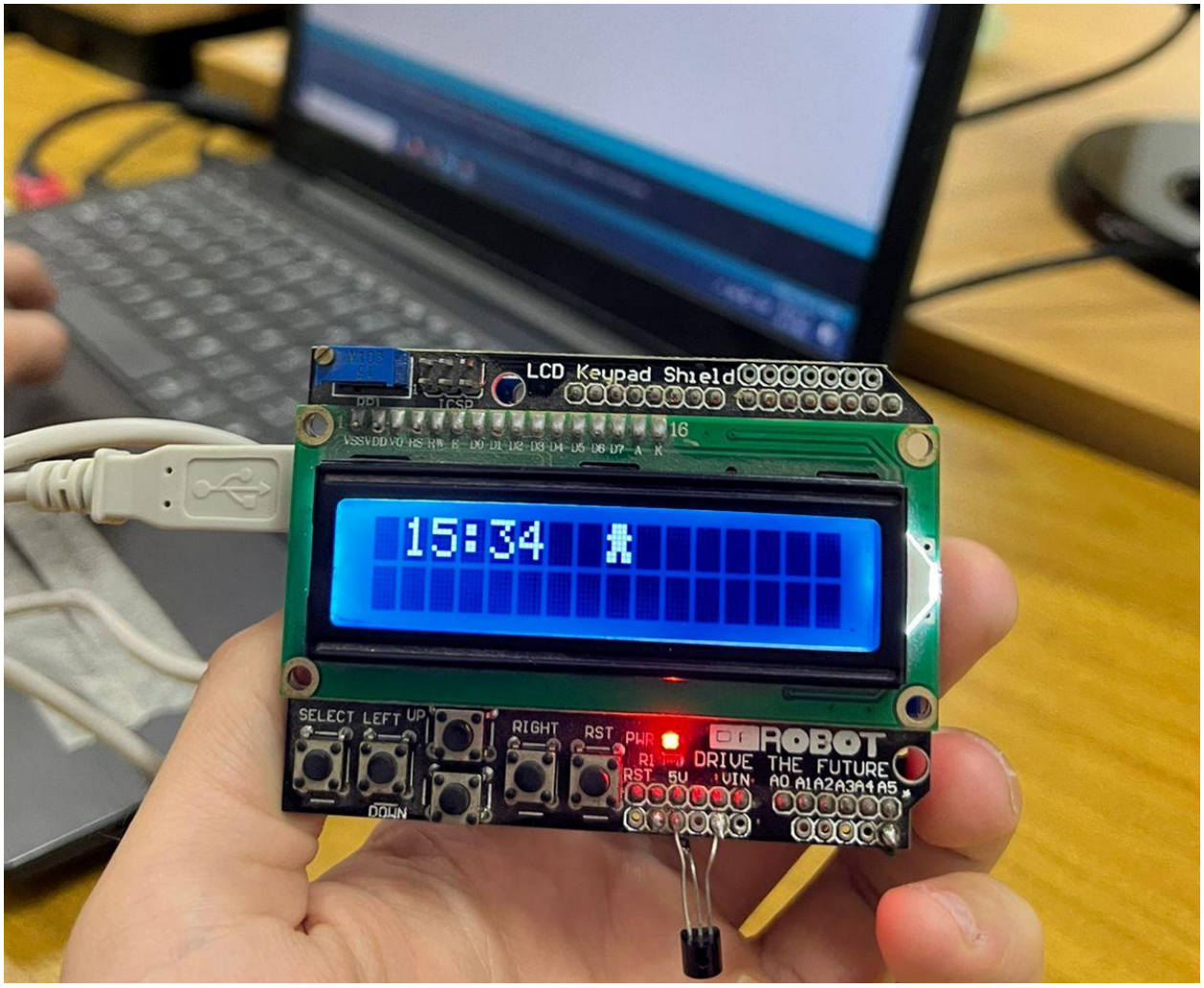
Uno de los días que realizamos pruebas en el laboratorio con un arduino real, al conectar el lcd al arduino tuvimos un problema con la configuración de los pin. Ya que dependiendo el modelo del LCD, van a configurarse unos o otros pines. Esto hacía que no se viera nada de lo que queríamos en la pantalla y que aparezca encendida todo el primer renglón de la misma.



Lo solucionamos cambiando los pines que se configuran para ese modelo de LCD en particular.

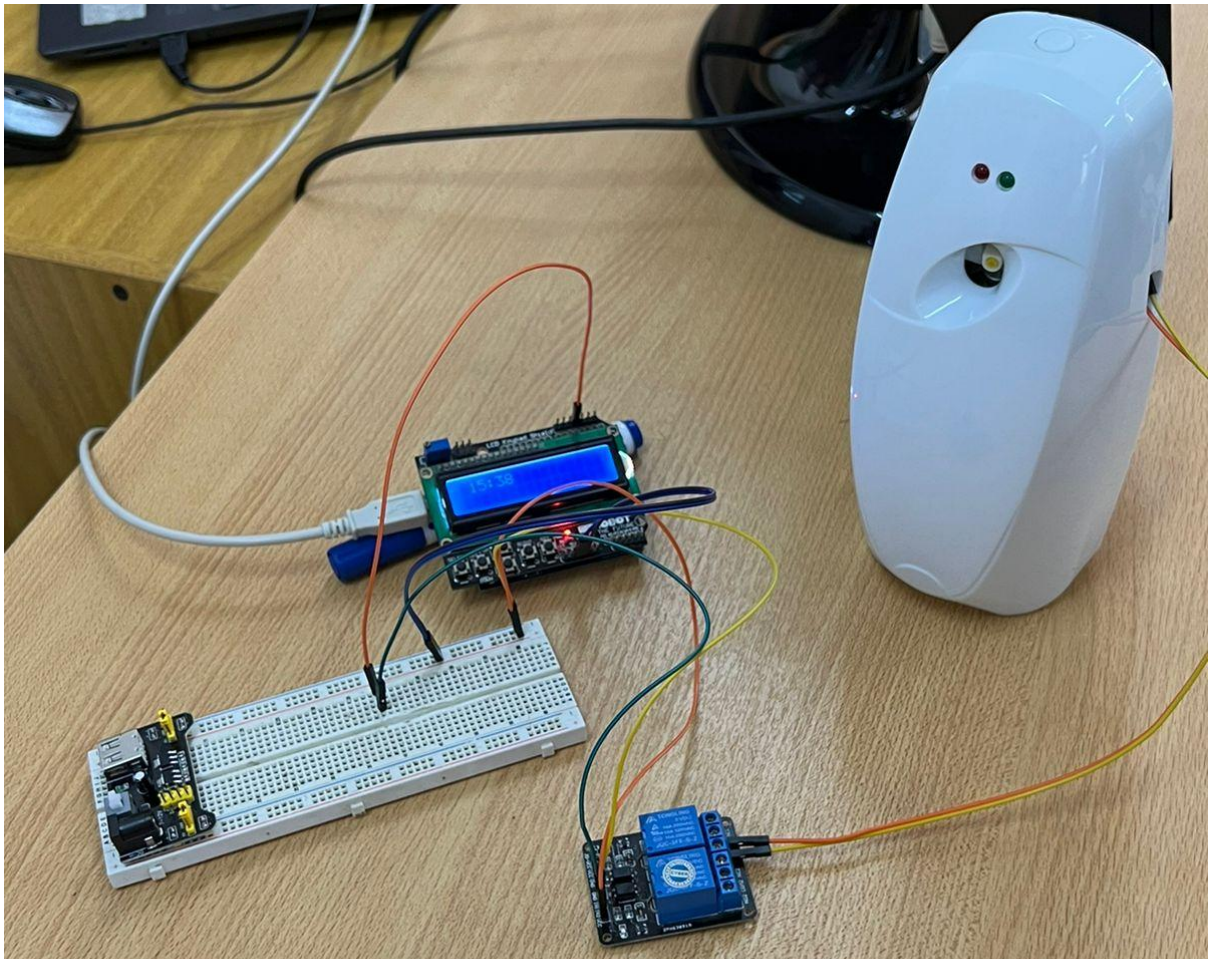
```
LiquidCrystal lcd_1(8,9,4,5,6,7);
```

Cambiado eso, la pantalla comenzó a funcionar correctamente y se muestra la hora y el dibujo.



El dia de la presentación del trabajo en el laboratorio, cambiamos los valores de salida de la salida digital, lo que estaba en LOW lo pusimos en HIGH y viceversa ya que el aromatizador se activa cuando la salida está en LOW y se desactiva cuando está en HIGH.

Finalmente hicimos la prueba. Ingresamos la hora y minutos separados por un espacio y la misma se mostró en el display. Luego introdujimos un intervalo de 1 minuto para que el aromatizador se active luego de pasado ese tiempo. Al pasar 1 minuto el aparato rocío el perfume, apareció un dibujo en la pantalla (que desapareció luego de un segundo) y la hora se incrementó en un minuto. El programa funcionaba según lo esperado. En el repositorio de GitHub se adjunta el video de la prueba.



Adjunto repositorios de GitHub de los 3 integrantes:

https://github.com/Ferkoss/TPI-SPD

https://github.com/InakiCabrera/TPSPDARDUINO.git

https://github.com/lucademinas/TPSPD-LucaDeminas.git