**1. Nombre del proyecto:** Control del nivel del líquido en un deposito

**2. Objetivo**

Controlar el llenado y vaciado de un depósito a través de Arduino y FPGA para mantener el agua en un nivel fijo.

**Objetivo especifico**

Utilizar un módulo relé con optoacoplador para proteger la placa FPGA/ARDUINO del circuito en general.

**3. Descripción (explicación)**

El siguiente ejercicio plantea a través de la lógica y electrónica digital un proceso que controla el nivel del líquido en un depósito el cuál es desarrollado a partir de sus compuertas universales, tablas de verdad y simplificación para desarrollarlo a través del microcontrolador Arduino y utilizando las tarjetas FPGA con el lenguaje de hardware VHDL.

**Planteamiento**

Se tiene un tanque o depósito que parte del estado inicial “totalmente vacío”. En donde, se habilita la bomba no sumergible (superior) y se deshabilita la bomba sumergible (inferior) hasta llenarse el tanque al nivel requerido.

Este proceso de llenado continua hasta que se active el primer sensor C.

En este momento, se pasa al estado “normal” activándose entonces ambas bombas (superior e inferior) manteniendo un nivel de agua constante.

Sí por alguna circunstancia externa (lluvia) se eleva el nivel del agua hasta activar al sensor B, se pasa al “estado lleno” y se cerraría la bomba no sumergible (superior) manteniéndose activa solamente la bomba sumergible (inferior) hasta volver nuevamente al “estado normal”.

Si por alguna razón se activa el sensor A, el sistema pasa al “estado de alarma” el cual se activa una alarma y las bombas se comportan de manera similar al “estado lleno” (bomba superior desactivada y bomba inferior activada)

**Variables de entrada:**

Sensor A, Sensor B, Sensor C.

**Variables de salida:**

Bomba No sumergible, Bomba sumergible, Alarma.

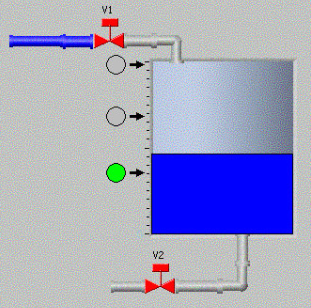
**Funcionamiento**

El algoritmo de control implementado se puede resumir de la forma siguiente:

* Si el líquido llega al nivel de vacío, se cierra la bomba de salida y se mantiene abierta la de entrada.
* En condiciones de llenado normal, las bombas E y S se encuentran abiertas.
* Si el líquido llega al nivel de lleno, se cierra la bomba de entrada y se mantiene abierta la de salida.
* Si por cualquier circunstancia, por ejemplo, lluvia, se llegara al nivel de alarma, se deberá cerrar la bomba de entrada y abrir la de salida mientras suena la alarma. Esta situación se mantendrá hasta que el tanque llegue al estado de normal.

**Proceso**

Se observa el esquema del depósito con dos bombas una para llenado y otra para vaciado. Además de estos elementos, también se dispone de tres sensores de nivel para controlar el nivel de llenado del depósito.



**Tabla de verdad:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ENTRADAS | | | SALIDAS | | |
| SA | SB | SC | BE | BS | A |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

En donde:

|  |  |  |  |
| --- | --- | --- | --- |
| BE | BS | A |  |
| 1 | 0 | 0 | VACIO |
| 1 | 1 | 0 | NORMAL |
| 0 | 1 | 0 | LLENO |
| 0 | 1 | 1 | ALARMA |

**Fórmulas de salidas**

Salida de la bomba de entrada (superior):

BE =

BE =

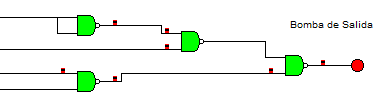


A  
B

Salida de la bomba de salida (inferior):

BS =

BS =



A

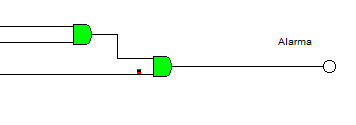
C

C

B

Salida de alarma:

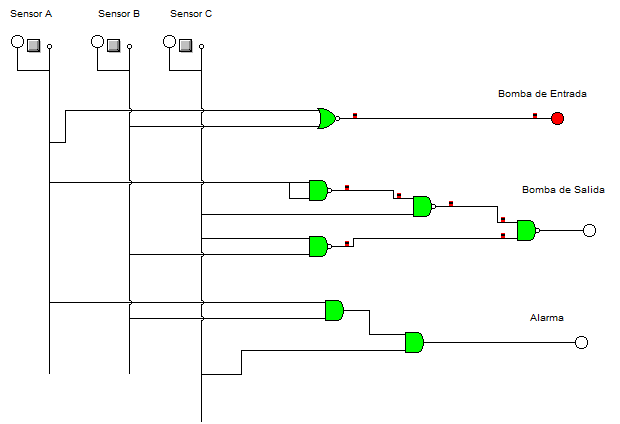
A =



A  
B

C

Finalmente, se tiene el circuito lógico completo:



**4. Códigos**

El funcionamiento básico del programa en VHDL/Arduino es el de mantener el nivel de líquido del depósito entre los niveles adecuados valiéndose de la información de los sensores y activando y desactivando tanto la bomba de entrada como la bomba de salida.

Los elementos que forman el proceso que se desea controlar son el depósito de líquidos, tres sensores de nivel de líquidos, dos bombas y la FPGA o Arduino. La FPGA/Arduino se utiliza para recibir como entradas las señales procedentes de los tres sensores de nivel y para controlar la apertura y cierre de las dos bombas.

**4.1 Código en VHDL**

A continuación, se presenta la versión del proyecto en VHDL el cual funciona de la misma forma que en Arduino.

1. library IEEE;
2. use IEEE.STD\_LOGIC\_1164.ALL;
3. entity TNQ1 is
4. Port ( SENS : in  STD\_LOGIC\_VECTOR (2 downto 0);
5. Ve : out  STD\_LOGIC;
6. Vs : out  STD\_LOGIC;
7. A : out  STD\_LOGIC);
8. end TNQ1;
9. architecture Behavioral of TNQ1 is
10. begin
11. process (SENS)
12. begin
13. if (SENS ="000") then
14. Ve<='1';
15. Vs<='0';
16. A<='0';
18. elsif (SENS ="001") then
19. Ve<='1';
20. Vs<='1';
21. A<='0';
23. elsif (SENS ="010") then
24. Ve<='0';
25. Vs<='0';
26. A<='0';
28. elsif (SENS ="011") then
29. Ve<='0';
30. Vs<='1';
31. A<='0';
33. elsif (SENS ="100") then
34. Ve<='0';
35. Vs<='0';
36. A<='0';
38. elsif (SENS ="101") then
39. Ve<='0';
40. Vs<='0';
41. A<='0';
43. elsif (SENS ="110") then
44. Ve<='0';
45. Vs<='0';
46. A<='0';
48. else
49. Ve<='0';
50. Vs<='1';
51. A<='1';
53. end if;
55. end process;
56. end Behavioral;

A continuación, se tiene el código Test bench que sirve para la simulación:

1. LIBRARY ieee;
2. USE ieee.std\_logic\_1164.ALL;
4. -- Uncomment the following library declaration if using
5. -- arithmetic functions with Signed or Unsigned values
6. --USE ieee.numeric\_std.ALL;
8. ENTITY TNQ1\_TB IS
9. END TNQ1\_TB;
11. ARCHITECTURE behavior OF TNQ1\_TB IS
13. -- Component Declaration for the Unit Under Test (UUT)
15. COMPONENT TNQ1
16. PORT(
17. SENS : IN  std\_logic\_vector(2 downto 0);
18. Ve : OUT  std\_logic;
19. Vs : OUT  std\_logic;
20. A : OUT  std\_logic
21. );
22. END COMPONENT;

25. --Inputs
26. signal SENS : std\_logic\_vector(2 downto 0) := (others => '0');
28. --Outputs
29. signal Ve : std\_logic;
30. signal Vs : std\_logic;
31. signal A : std\_logic;
32. -- No clocks detected in port list. Replace <clock> below with
33. -- appropriate port name
35. --constant <clock>\_period : time := 10 ns;
37. BEGIN
39. -- Instantiate the Unit Under Test (UUT)
40. uut: TNQ1 PORT MAP (
41. SENS => SENS,
42. Ve => Ve,
43. Vs => Vs,
44. A => A
45. );
47. -- Clock process definitions
48. --   <clock>\_process :process
49. --   begin
50. --              <clock> <= '0';
51. --              wait for <clock>\_period/2;
52. --              <clock> <= '1';
53. --              wait for <clock>\_period/2;
54. --   end process;

57. -- Stimulus process
58. stim\_proc: process
59. begin
60. SENS<="000", "001" after 100 ns,"010" after 200 ns,"011" after 300 ns,"100" after 400 ns,
61. "101" after 500 ns,"011" after 600 ns,"111" after 700 ns;
63. -- hold reset state for 100 ns.
64. wait for 400 ns;
66. --wait for <clock>\_period\*10;
68. -- insert stimulus here
70. wait;
71. end process;
73. END;

**4.2 Código en Arduino.**

A continuación, se muestra la lógica combinacional implementada en Arduino en donde se utiliza una placa de Arduino mega:

**Versión 1**

Aquí se detalla el código en Arduino solamente de las compuertas que se ha implementado anteriormente.

1. //Entradas (sensores)
2. int sA =22; //Sensor Parte superior (ESTADO ALARMA)
3. int sB = 24; //Sensor del medio (ESTADO LLENO)
4. int sC = 26; //Sensor ultimo (ESTADO NORMAL)
6. //Salidas (Bombas y alarma)
8. int bNoSumergible = 28; //El relé activa bomba No Sumergible/ValVula de entrada
9. int bSumergible = 30; //El relé activa bomba Sumergible/Valvula de salida
10. int alarma = 32; //Alarma suena cuando llega al Sensor A
12. //Variables
13. int estadoNormal;
14. int estadoLleno;
15. int estadoAlarma;
16. void setup() {
18. pinMode(sA, INPUT);
19. pinMode(sB, INPUT);
20. pinMode(sC, INPUT);
22. pinMode(bSumergible, OUTPUT);
23. pinMode(bNoSumergible, OUTPUT);
24. pinMode(alarma, OUTPUT);
25. }

28. void loop() {
30. estadoNormal = digitalRead(sC);
31. estadoLleno = digitalRead(sB);;
32. estadoAlarma = digitalRead(sA);
34. //Sensor A          Sensor B           Sensor C
35. if(estadoAlarma==0 && estadoLleno ==0 && estadoNormal==0){
37. delay(3000); //retraso para esperar y ver como se activa la bomba no sumergible.
39. while(estadoNormal==0){
41. digitalWrite(bNoSumergible, HIGH); //La B.N.sumergi comienza activa.
42. /\*Cuando el agua alcance al sensor C, se guardara en estadonormal el
43. numero 1, y saldrá del blucle while.
44. \*/
45. estadoNormal = digitalRead(sC);
46. }
47. digitalWrite(bNoSumergible, LOW); //finalmente apago la bomba Nsumergi. por que ya llego al sC.
48. digitalWrite(bSumergible, LOW);
49. digitalWrite(alarma, LOW);
51. }else if(estadoAlarma==0 && estadoLleno ==0 && estadoNormal==1){
53. delay(3000); //retraso para esperar y ver como se activan las 2 bombas al mismo tiempo
54. //y el agua finalmente se mantiene constante en el sensor C=1
55. while( estadoLleno ==0 && estadoNormal ==1){
57. digitalWrite(bNoSumergible, HIGH);
58. digitalWrite(bSumergible, HIGH);
60. //estadoAlarma = digitalRead(sA);
61. estadoLleno = digitalRead(sB);
62. estadoNormal = digitalRead(sC);
63. }
65. digitalWrite(bNoSumergible, LOW);
66. digitalWrite(bSumergible, LOW);
67. digitalWrite(alarma, LOW);
69. }else if(estadoAlarma==0 && estadoLleno ==1 && estadoNormal==1){
70. delay(3000); //retraso para esperar y ver como se activa la bomba sumergible.
72. while(estadoAlarma==0 &&estadoLleno==1 && estadoNormal==1){
74. digitalWrite(bSumergible, HIGH); //La B.sumergi se activa para que se valla el agua demás.
75. /\*Cuando el agua baje lo suficiente para que el sensor B (estado lleno) vuelva a 0. El
76. bucle termina.
77. \*/
78. estadoLleno = digitalRead(sB);
79. estadoNormal =digitalRead(sC);
80. estadoAlarma= digitalRead(sA);
81. }
82. digitalWrite(bNoSumergible, LOW);
83. digitalWrite(bSumergible, LOW);//finalmente apago la bomba sumergi. por que ya llego al sC y sB=0.
84. digitalWrite(alarma, LOW);
86. }else if(estadoAlarma==1 && estadoLleno ==1 && estadoNormal==1){
88. delay(3000); //retraso para esperar y ver como se activa la bomba NOsumergible.
90. while(estadoAlarma==1 ){
92. digitalWrite(bSumergible, HIGH); //La B.sumergi se activa para que se valla el agua demás.
93. /\*Cuando el agua baje lo suficiente para que el sensor A (estado alarma) vuelva a 0. El
94. bucle termina.
95. \*/
96. digitalWrite(alarma, HIGH);    //La alarma se activa mientras estadoAlarma(sA)=1
97. estadoAlarma = digitalRead(sA);
98. }
99. digitalWrite(bNoSumergible, LOW);
100. digitalWrite(bSumergible, LOW);//
101. digitalWrite(alarma, LOW);
102. }
103. }

A continuación, se presenta el código de la alarma que se utiliza, en este caso es un buzzer pasivo y es un código aparte:

1. /\*
2. Conexión
3. Buzzer -------- Arduino Mega
4. Terminal + = PIN 8
5. Terminal - = GND
6. Agregar resistencia de 100 Ohm si suena alto, incrementar resistencia si aún suena alto.
7. Quitar resistencia si no suena bajo.
9. \*/
11. //Declaración de las variables
12. int pBuzzer = 40; // pin del buzzer pasivo
13. int cuarto = 1000 / 4; // tiempo de 1/4 de la nota
14. int octavo = 1000 / 8; // tiempo de 1/8 de la nota
15. double pausa = 1.30; // pausar el 30% de la nota
17. // La frecuencia min que puede producir el Arduino Uno es 31Hz.
18. // La frecuencia max que puede producir el Arduino Uno es 65,525Hz.
19. int NOTA\_C4 = 262; // Frecuencia de la nota C4 es 262Hz
20. int NOTA\_A4 = 440; // Frecuencia de la nota A4 es 440Hz
21. int NOTA\_E5 = 659; // Frecuencia de la nota E5 es 659Hz
23. void setup() {
24. // put your setup code here, to run once:
25. pinMode(pBuzzer, OUTPUT);
26. }
28. void loop() {
29. // put your main code here, to run repeatedly:
30. tone(pBuzzer, NOTA\_C4, cuarto); //un cuarto de la nota C4.
31. delay(cuarto\*pausa); // silenciar por una duración de la nota previa + 30% de la duración de esa nota
32. tone(pBuzzer, NOTA\_A4, octavo); //un cuarto de la nota A4.
33. delay(octavo\*pausa); // silenciar por una duración de la nota previa + 30% de la duración de esa nota
34. tone(pBuzzer, NOTA\_E5, cuarto); //un cuarto de la nota E5.
35. delay(cuarto\*pausa); // silenciar por una duración de la nota previa + 30% de la duración de esa nota
36. }

**Versión 2**

Aquí se tiene el código final utilizado en el proyecto con las líneas de la alarma introducidas en el código general, además se corrigió errores tales como el LOW y HIGH de las bombas el cual en LOW significan que están activadas y en HIGH significan que están desactivadas.

1. //Entradas (sensores)
2. int sA = 22; //Sensor Parte superior (ESTADO ALARMA)
3. int sB = 24; //Sensor del medio (ESTADO LLENO)
4. int sC = 26; //Sensor ultimo (ESTADO NORMAL)

7. //Salidas (Bombas y alarma)
9. int bNoSumergible = 28; //El relé activa bomba No Sumergible/Bomba de entrada.
10. int bSumergible = 30; //El relé activa bomba Sumergible/Bomba de salida.
11. int alarma = 32; // pin del buzzer pasivo
13. //Variables para controlar cada sensor
14. int estadoNormal; //sA
15. int estadoLleno; //sC
16. int estadoAlarma; //sC
18. //------------------------------------------------------------------
19. //-----variables del buzzer pasivo (alarma)
20. int cuarto = 1000 / 4; // tiempo de 1/4 de la nota
21. int octavo = 1000 / 8; // tiempo de 1/8 de la nota
22. double pausa = 1.30; // pausar el 30% de la nota
24. // La frecuencia min que puede producir el Arduino Uno es 31Hz.
25. // La frecuencia max que puede producir el Arduino Uno es 65,525Hz.
26. int NOTA\_C4 = 262; // Frecuencia de la nota C4 es 262Hz
27. int NOTA\_A4 = 440; // Frecuencia de la nota A4 es 440Hz
28. int NOTA\_E5 = 659; // Frecuencia de la nota E5 es 659Hz
29. //-----------------------------------------------------------------
31. void setup() {
33. pinMode(sA, INPUT);
34. pinMode(sB, INPUT);
35. pinMode(sC, INPUT);
37. pinMode(bSumergible, OUTPUT);
38. pinMode(bNoSumergible, OUTPUT);
39. pinMode(alarma, OUTPUT);
41. Serial.begin(9600); //abre el puerto de serie para la transmision de ceros
42. }

45. void loop() {
47. estadoNormal = digitalRead(sC);
48. estadoLleno = digitalRead(sB);;
49. estadoAlarma = digitalRead(sA);
51. //Sensor A                Sensor B           Sensor C
52. if (estadoAlarma == 0 && estadoLleno == 0 && estadoNormal == 0) {
54. // delay(1000); //retraso para esperar y ver como se activa la bomba no sumergible (de salida).
56. while (estadoNormal == 0) {
58. digitalWrite(bSumergible, HIGH); //Mi bomba de salida va a estar desactivada
59. digitalWrite(bNoSumergible, LOW); //La B.N.sumergible (de entrada) está activa.
60. Serial.println("Estado 0 0 0 bomba no sumergible accionada.");
61. /\*Cuando el agua alcance al sensor C, se guardara en estadonormal el
62. numero 1, y saldrá del blucle while.
63. \*/
64. estadoNormal = digitalRead(sC);
65. }
66. digitalWrite(bNoSumergible, LOW); // Bomba de llenado (de entrada)continua prendida.
67. digitalWrite(bSumergible, LOW); //Bomba de salida SE ACTIVA
68. digitalWrite(alarma, LOW); //Alarma apagada
70. } else if (estadoAlarma == 0 && estadoLleno == 0 && estadoNormal == 1) {
72. // delay(1000); //retraso para esperar y ver como se activan las 2 bombas al mismo tiempo
73. //y ver como el agua se mantiene constante en el sensor C=1.
74. while ( estadoLleno == 0 && estadoNormal == 1) {
76. digitalWrite(bNoSumergible, LOW); //Bomba de salida activada
77. digitalWrite(bSumergible, LOW); //Bomba de entrada activada
78. Serial.println("Estado 0 0 1 bomba no sumergible y sumergible accionadas.");
79. //estadoAlarma = digitalRead(sA);
80. estadoLleno = digitalRead(sB);
81. estadoNormal = digitalRead(sC);
82. }
84. digitalWrite(bNoSumergible, HIGH); //Se apagan
85. digitalWrite(bSumergible, HIGH);//Se apagan
86. digitalWrite(alarma, LOW); //Sigue apagado

89. } else if (estadoAlarma == 0 && estadoLleno == 1 && estadoNormal == 1) {
90. //delay(1000); //retraso para esperar y ver como se activa la bomba sumergible ( de salida).
92. while (estadoAlarma == 0 && estadoLleno == 1 && estadoNormal == 1) {
94. digitalWrite(bSumergible, LOW); //La B.sumergi se activa para que se valla el agua que está demás.
95. /\*Cuando el agua baje lo suficiente para que el sensor B (estado lleno) vuelva a 0. El
96. bucle termina.
97. \*/
98. Serial.println("Estado 0 1 1 bomba sumergible accionada.");
99. estadoLleno = digitalRead(sB);
100. estadoNormal = digitalRead(sC);
101. estadoAlarma = digitalRead(sA);
102. }
103. digitalWrite(bNoSumergible, HIGH); //Bomba de entrada se mantiene desactivada
105. //----------------------------
106. if(estadoLleno == 0){
108. delay(80000); //esperar hasta que el agua baje al nivel sC
109. }
110. //---------------------------
112. digitalWrite(bSumergible, HIGH);//finalmente apago la bomba sumergi. por que ya llego al sC ==1 y sB==0.
113. digitalWrite(alarma, LOW); //Alarma sigue apagada
115. } else if (estadoAlarma == 1 && estadoLleno == 1 && estadoNormal == 1)  {
117. //delay(1000); //retraso para esperar y ver como se activa la bomba NOsumergible.
119. while (estadoAlarma == 1 ) {
121. digitalWrite(bSumergible, LOW); //La B.sumergi se activa para que se valla el agua demás.
122. /\*Cuando el agua baje lo suficiente para que el sensor A (estado alarma) vuelva a 0. El
123. bucle termina.
124. \*/
125. digitalWrite(alarma, HIGH);    //La alarma se activa mientras estadoAlarma(sA)=1
126. estadoAlarma = digitalRead(sA);
128. Serial.println("Estado 1 1 1 bomba sumergible y alarma accionadas.");
130. // Melodía del buzzer
131. tone(alarma, NOTA\_C4, cuarto); //un cuarto de la nota C4.
132. delay(cuarto \* pausa); // silenciar por una duración de la nota previa + 30% de la duración de esa nota
133. tone(alarma, NOTA\_A4, octavo); //un cuarto de la nota A4.
134. delay(octavo \* pausa); // silenciar por una duración de la nota previa + 30% de la duración de esa nota
135. tone(alarma, NOTA\_E5, cuarto); //un cuarto de la nota E5.
136. delay(cuarto \* pausa); // silenciar por una duración de la nota previa + 30% de la duración de esa nota
137. //---------------------------------------------------------------------------------------
138. }
140. //una vez sale del bucle, quiere decir que el sensor a volvió a 0, sA==0 entonces:
141. digitalWrite(bNoSumergible, HIGH); //Bomba de entrada sigue desactivada como antes
142. // digitalWrite(bSumergible, HIGH);//
143. digitalWrite(alarma, LOW); //Alarma finalmente se apaga por que sA==0.
144. delay(80000); //esperar hasta que el agua baje al nivel sC
146. }
147. }

**5. Construcción**

**Materiales:**

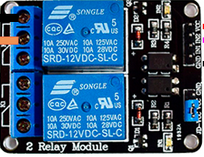
* 1 bomba sumergible y otra no sumergible.



* 3 sensores de nivel de agua.



* 3 metros de cable y cables macho-hembra para Arduino.
* 1 placa Arduino mega.
* 1 modulo relé de 2 canales (para activar las bombas).



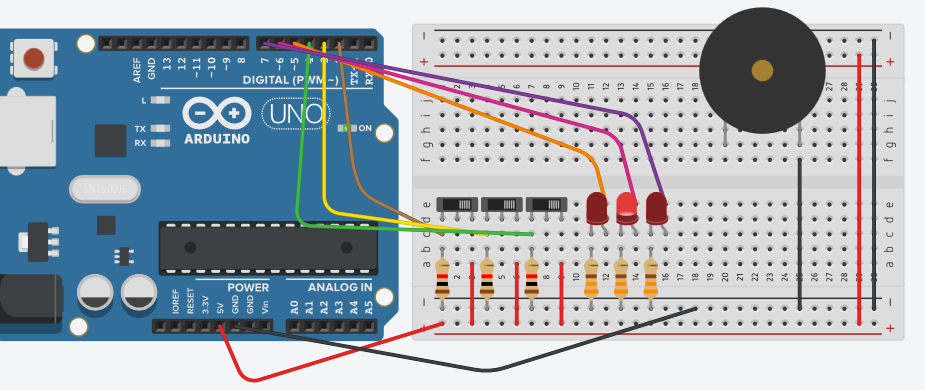
* Protoboard.
* 3 resistencias de 1Kohm.
* Fuente de 12 voltios (por que nuestra bomba no sumergible funciona a12v).
* 1 tanque de plástico cualquier tamaño.
* 1 metro de tubo plástico transparente de peceras.

**5.1 Circuito:**

Los leds de izquierda a derecha representan la bomba no sumergible (superior), la bomba sumergible (inferior) y, por último, la alarma.

Los interruptores de izquierda a derecha son el sensor A, sensor B y sensor C y están conectados a través de una resistencia pull down de 1k.

Ejemplo: La siguiente figura muestra el estado 0 0 0. El cual, la bomba no sumergible esta activada hasta que se active el sensor C (interruptor).





Cabe recalcar que en la práctica los interruptores se reemplazan por los sensores de nivel, el primer led se reemplaza por una bomba no sumergible que funciona a 12 voltios, conectada a través de un relé de 5v. El segundo led se reemplaza por una bomba sumergible que funciona a 5v del Arduino. Y el último led se reemplaza por el buzzer que está ubicado en el protoboard que funciona a 5v.

**6.1 Problemas**

En el desarrollo de este proyecto surgió el problema de la velocidad del flujo de agua de las bombas. Una era más rápida por lo que se utilizó tiempos de retraso en la programación de Arduino para controlar su llenado y vaciado.

**7. Conclusiones**

* El código en Arduino resulto más fácil de escribirlo que el de VHDL debido al entorno y teoría que hay en la web. El código de VHDL se tornó difícil realizarlo debido a su simulación y quemado en la tarjeta.
* Se utilizó tiempos de retraso para que el nivel del agua vuelva al sensor c y no halla errores de medición.

**8. Recomendaciones**

* Las resistencias de pull down deben ser de 1Kohm.

**9.Bibliografía:**

Autoría propia.