INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
Ingeniería en Comunicaciones y Electrónica

Reporte de Proyecto final.

Campos Castillo Brandon Alexis

Microcontroladores

Sensor de Distancias menores a 10 cm.

Material necesario.

-Raspberry pi (cualquier modelo a partir de la Zero)
-Sensor ultrasónico.
-Resistor 330Ω
-Resistor 470Ω
-TM-1638
-Relay Module
-Ventilador

Marco Teórico.

Raspberry PI.

La Raspberry Pi es una serie de ordenadores de placa reducida, ordenadores de placa única u ordenadores de placa simple (SBC) de bajo costo desarrollado en el Reino Unido por la Raspberry Pi Foundation, con el objetivo de poner en manos de las personas de todo el mundo el poder de la informática y la creación digital. Si bien el modelo original buscaba la promoción de la enseñanza de informática en las escuelas, este acabó siendo más popular de lo que se esperaba, hasta incluso vendiéndose fuera del mercado objetivo para usos como robótica.

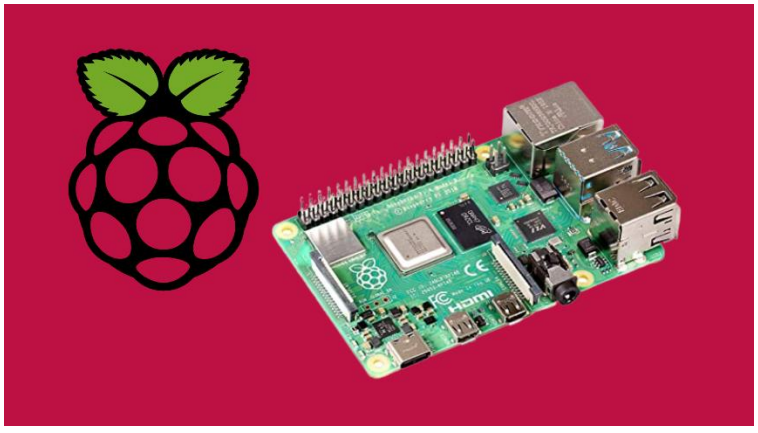

Imagen 1. "Raspberry Pi".

Modulo Display Siete Segmentos Tm1638 X8

El módulo de visualización TM1638 de 7 segmentos es un dispositivo utilizado en el desarrollo de proyectos electrónicos desarrollados a partir de plataformas de creación de prototipos, incluido Arduino y Raspberry Pi.

Se puede usar en varias aplicaciones, como, por ejemplo, la visualización de mensajes numéricos y de texto, para demostrar sistemas de conteo, cronogramas, códigos de comando, entre otras aplicaciones.

Extremadamente funcional, también tiene 8 LED rojos, una pantalla numérica con 8 dígitos y 8 botones de ajuste, que pueden usarse para los más diversos propósitos, dependiendo de la programación.

Los LED y botones Módulo Pantalla 7 segmentos TM1638 se puede usar para proporcionar valores o comentarios en respuesta a los mensajes de error que se muestran en la pantalla.

Para su funcionamiento, utiliza un controlador TM1638 que tiene varios ejemplos de programación disponibles en Internet, por lo que es una herramienta fácil de usar.

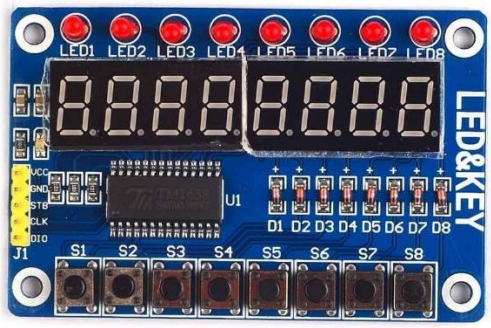

Imagen 2. "TM-1638"

Sensor Ultrasónico.

Como su nombre lo indica, los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción.

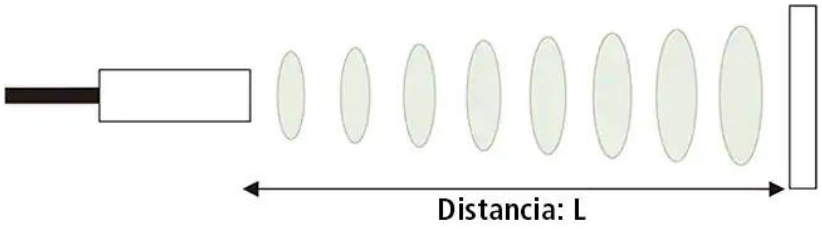

Imagen 3. "Representación de lo que hace un Sensor ultrasónico"

Un sensor óptico tiene un transmisor y receptor, mientras que un sensor ultrasónico utiliza un elemento ultrasónico único, tanto para la emisión como la recepción. En un sensor ultrasónico de modelo reflectivo, un solo oscilador emite y recibe las ondas ultrasónicas, alternativamente. Esto permite la miniaturización del cabezal del sensor.

La distancia se puede calcular con la siguiente fórmula:

$$Distancia\ L\ =\ 1/2\ \times\ T\ \times\ C$$

Donde L es la distancia, T es el tiempo entre la emisión y la recepción, y C es la velocidad del sonido. (El valor se multiplica por 1/2 ya que T es el tiempo de recorrido de ida y vuelta).

Relay Module.

Un relé es un interruptor, más frecuentemente electromagnético, que utiliza una pequeña corriente para accionar un circuito mayor. Básicamente, se aplica una señal en la entrada que enciende otro circuito conectado en la salida, sin necesidad de supervisión humana.

Los relés funcionan según el principio de inducción electromagnética, lo que implica que, si la corriente pasa a través de una bobina que se enrolla alrededor de un trozo de metal, entonces se producirá un campo magnético por la corriente, convirtiendo el núcleo de metal en un electroimán.

Si miramos dentro de un relé, encontraremos un electroimán que tiene una armadura conectada a los contactos que conectan el terminal de entrada y salida cuando la corriente eléctrica pasa a través de la bobina. El embalaje exterior de un relé tiene cinco clavijas, dos de las cuales se conectan a cada lado de la bobina y al aplicar un voltaje a estas clavijas se energiza la bobina. Los tres pines restantes son - Normalmente Abierto (NO), Normalmente Cerrado (NC) y Común (COM).

Originalmente, la armadura conecta los contactos COM y NC, pero cuando la corriente pasa a través del electroimán y carga la bobina, crea un campo magnético que atrae a la armadura y, por lo tanto, la armadura pasa de la posición normalmente cerrada a la posición normalmente abierta, impulsando el circuito más grande conectado en el pin NO. Este es el principio básico de funcionamiento de un relé. Ahora entendamos qué es un módulo de relé.

Imagen 4. "Relay Module"

Conexiones.

El sensor Ultrasónico tiene 4 Pines de conexión (GND, ECHO, TRIG, VCC) cada uno ira conectado a un pin diferente, y en el caso de (GND y VCC) irán conectados a una línea de la protoboard que va a compartir la tierra y la corriente con otros dispositivos.

Los pines para el sensor ultrasónico serian:
TRIG pin #18
ECHO pin #24.
NOTA: PARA EL ECHO SE LE DEBEN CONECTAR EL RESISTOR DE 330 OHMS DESPUES IRA LA CONEXIÓN AL PIN NUM 24 Y POR ÚLTIMO EN LA MISMA LINEA SE CONECTARÁ EL OTRO RESISTOR DE 470 OHMS A LA TIERRA.

Para la conexión de el sensor ultrasónico quedaría de la siguiente manera:

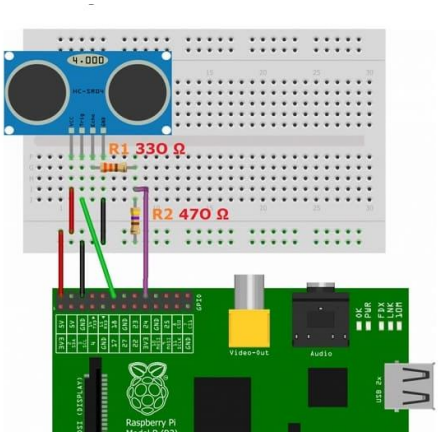

Imagen 5. "Conexión sensor ultrasónico"

Para la conexión del TM-1638.
El TM-1638 tiene 5 pines de conexión (VCC, GND, STB, DIO, CLK) para VCC Y GND se conectarán a corriente y tierra respectivamente y cada uno de los siguientes pines se conectarán de la siguiente manera.
STB pin #37
CLK pin #33
DIO pin #36

Y por último se usará el pin #23 y se le llamara OUT y será el que se conectara a el Relay Module donde se conectara el ventilador.

Las conexiones terminarían de la siguiente manera.

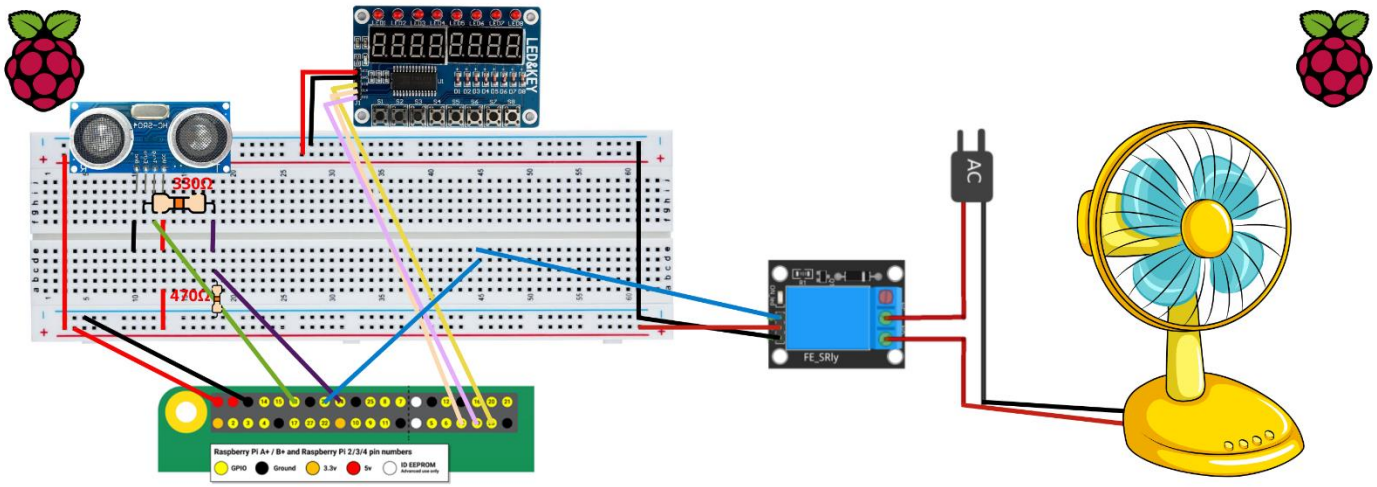

Imagen 6. "Conexiones totales."

DESCRIPCION GENERAL DE FUNCIONAMIENTO DE LA PRACTICA.

Se inician 2 terminales para el funcionamiento optimo del programa, y se iniciara el servidor en una de las terminales, y en la otra terminal se iniciará el programa, que esperara respuesta del servidor para iniciar el funcionamiento de el medidor de distancia.

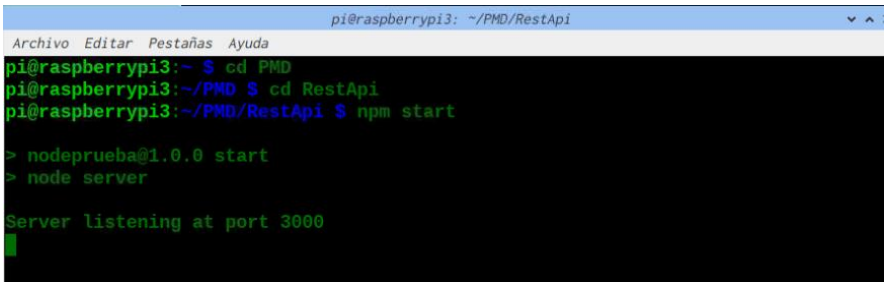

Imagen 7. "Servidor iniciado"

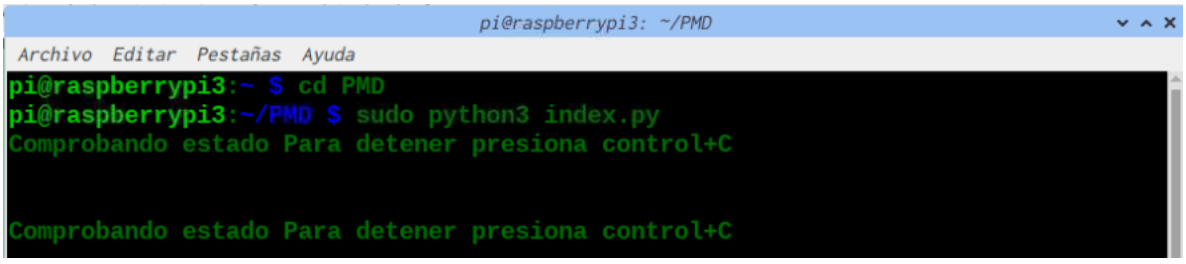

Imagen 8. "Programa en Espera"

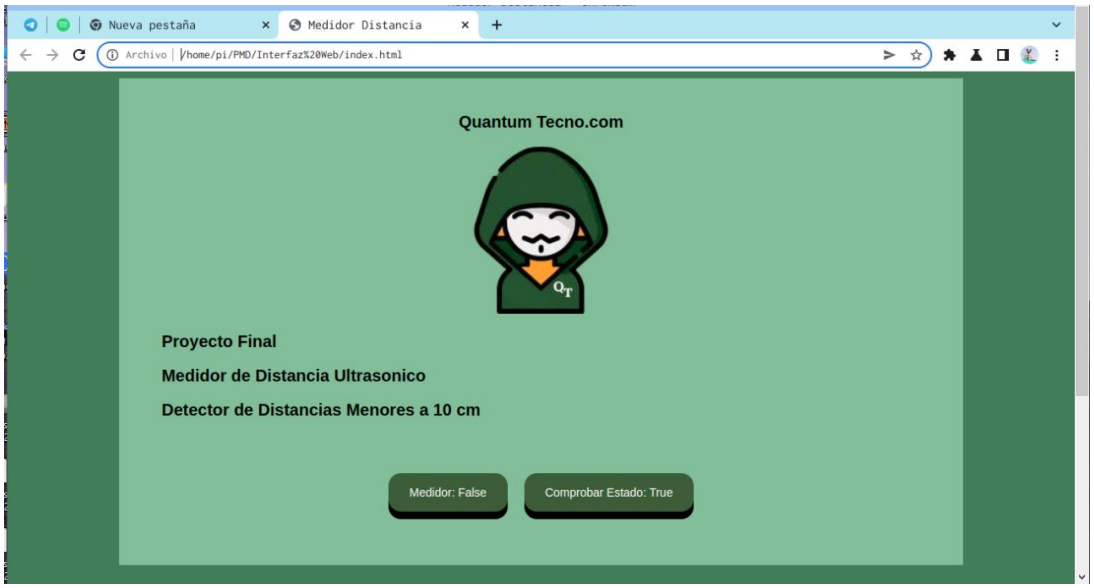

Imagen 9. "Página Web"

El botón "Comprobador de estado" se encarga de avisar que si se esta recibiendo información del servidor imprimiendo en el programa "Comprobando estado, Para detener presiona control + C"

El botón Medidor se encargara de activar el medidor de distancia mostrando las medidas en 2do plano y no en la pagina web.

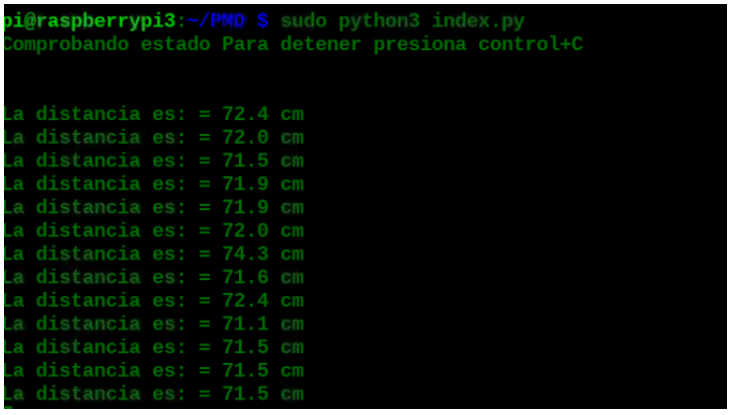

Imagen 10. "Medidas iniciadas"

Cuando el programa detecte que la distancia medida es menor a 10 cm mandara un mensaje a la consola y en el TM-1638 imprimirá "DIST-10" indicando que la distancia es menos a 10 cm al mismo tiempo se iniciara el ventilador gracias al Relay module.

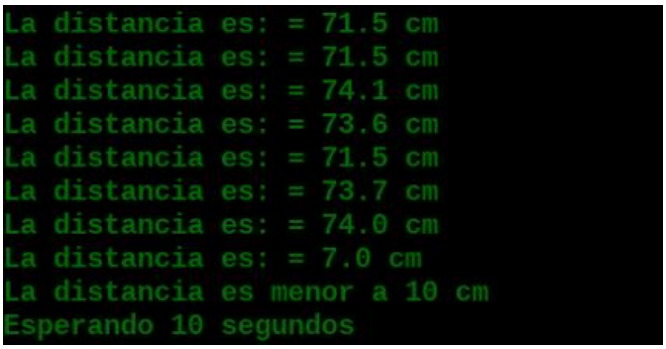

Imagen 11. "Distancia menor a 10 cm"

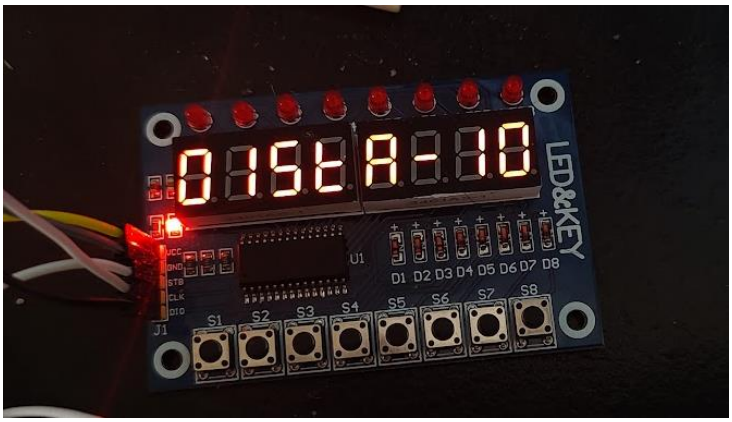

Imagen 12. "TM-1638 imprimiendo mensaje"

Imagen 13. "Ventilador encendido."

Para apagar será necesario, poner en "false" el botón medidor y presionar Control + C en el programa. Así el programa solo estará en espera de un cambio nuevamente en el

botón medidor.
Para finalizar el programa solo será necesario presionar Control + C nuevamente.

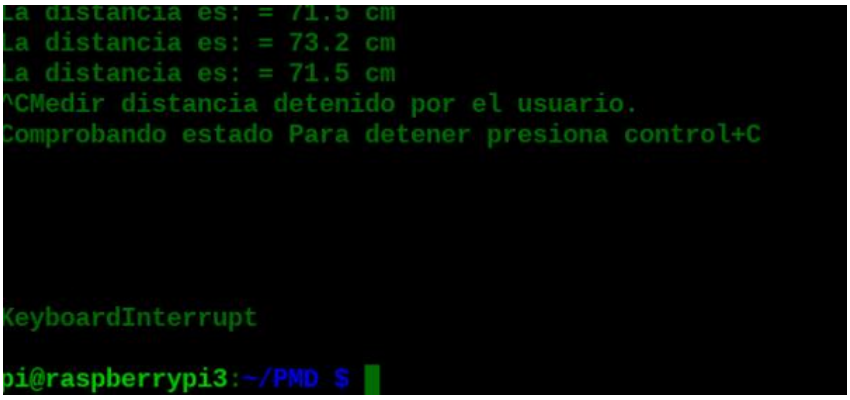

Imagen 14. "Programa Detenido."

Códigos.

Index.py

```
1. import json
2. import time
3. #Libraries
4. import RPi.GPIO as GPIO
5. import socket
6. from rpi_TM1638 import TMBoards
7. # Variable global para guardar el estado
8. estados = {
9.     "encenderLed": False,
10.         "encenderMotor": False
11.     }
12.     def actualiceConfig():
13.         global estados
14.
15.         with open('config.json',) as f:
16.             config = json.load(f)
17.             # Convertir de string a booleano
18.             config['encenderLed'] = True if config['encender
   Led'] == "True" else False
19.             config['encenderMotor'] = True if config['encend
   erMotor'] == "True" else False
20.
21.             estados = config
22.             f.close()
23.
24.     if __name__ == '__main__':
25.         while(1):
```

```
26.             actualiceConfig()
27.             if estados['encenderLed'] == True:
28.                 #TM Para imprimir en modilo tm-1638
29.                 DIO = 19 #pin 35
30.                 CLK = 13 #pin 33
31.                 STB = 26 #pin 37
32.                 TM = TMBoards(DIO, CLK, STB, 0)
33.                 TM.clearDisplay()
34.
35.
36.                 #GPIO Mode (BOARD / BCM)
37.                 GPIO.setmode(GPIO.BCM)
38.                 GPIO.setwarnings(False)
39.
40.                 #set GPIO Pins
41.                 GPIO_TRIGGER = 18
42.                 GPIO_ECHO = 24
43.
44.
45.                 #set GPIO direction (IN / OUT)
46.                 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
47.                 GPIO.setup(GPIO_ECHO, GPIO.IN)
48.
49.                 def distance():
50.                 # set Trigger to HIGH
51.                 #GPIO.output(GPIO_TRIGGER, True)
52.                     GPIO.output(GPIO_TRIGGER, GPIO.HIGH)
53.                 # set Trigger after 0.01ms to LOW
54.                     time.sleep(0.00001)
55.                     GPIO.output(GPIO_TRIGGER, False)
56.
57.                     StartTime = time.time()
58.                     StopTime = time.time()
59.
60.                 # save StartTime
61.                     while GPIO.input(GPIO_ECHO) == 0:
62.                         StartTime = time.time()
63.
64.                 # save time of arrival
65.                     while GPIO.input(GPIO_ECHO) == 1:
66.                         StopTime = time.time()
67.
68.                 # time difference between start and arrival
69.                     TimeElapsed = StopTime - StartTime
70.                 # multiply with the sonic speed (34300 cm/s)
71.                 # and divide by 2, because there and back
72.                     distance = (TimeElapsed * 34300) / 2
73.
74.                     return distance
75.                 if __name__ == '__main__':
```

```
                    try:
                        while True:
                            dist = distance()
                            print ("La distancia es: = %.1f cm" % dist)
                            y=10
                            if dist<y:

                                print ("La distancia es menor a 10 cm") #Imprime que la distancia es menor a 10 cm y lo manda al TM
                                print ("Esperando 10 segundos")
                                TM.segments[0]= 'D'
                                TM.segments[1]= 'I'
                                TM.segments[2]= 'S'
                                TM.segments[3]= 'T'
                                TM.segments[4]= 'A'
                                TM.segments[5]= '-'
                                TM.segments[6]= '1'
                                TM.segments[7]= '0'

                                GPIO.setup(23, GPIO.OUT)
                                GPIO.output(23, GPIO.HIGH)
                                time.sleep(10)
                                GPIO.output(23, GPIO.LOW)
                                TM.segments[0]= ' '
                                TM.segments[1]= ' '
                                TM.segments[2]= ' '
                                TM.segments[3]= ' '
                                TM.segments[4]= ' '
                                TM.segments[5]= ' '
                                TM.segments[6]= ' '
                                TM.segments[7]= ' '
                            time.sleep(1)
                # Reset by pressing CTRL + C
                    except KeyboardInterrupt:
                     print("Medir distancia detenido por el usuario. ")
                     TM.clearDisplay()
                     GPIO.cleanup()

            if estados['encenderMotor'] == True:
                print("Comprobando estado Para detener presiona control+C")
            print('\n')
            time.sleep(5)
```

server.js

```
1. const express = require ( 'express' ) ;
2. const fs = require ( 'fs' ) ;
3. var cors = requerir ( 'cors' )
4.
5. const app = express ( ) ;
6. puerto constante = 3000 ;
7.
8. aplicación uso ( express. json ( {
9.     tipo : '*/*' ,
10.     } ) ) ;
11.
12.     // Para obtener la informacion del archivo
13.     aplicación get ( '/' , cors ( ) , ( req , res ) => {
14.         // Lee el archivo
15.         sea
   rawdata = fs. readFileSync ( '../config.json' ) ;
16.         dejar configuración = JSON. analizar ( datos sin
   procesar ) ;
17.         // Responde con la información del archivo
18.         res. json ( configuración ) ;
19.     } ) ;
20.
21.     // Para modificar la bandera en el archivo
22.     aplicación publicación ( '/' , cors ( ) , ( req , res )
   => {
23.         dejar newConfig = req. cuerpo _
24.         fs. writeFileSync ( '../config.json' , JSON. stringi
   fy ( nueva configuración ) )
25.         res. json ( { mensaje : 'Archivo modificado con
   exito' } ) ;
26.     } ) ;
27.
28.     // Se pone en escucha el servidor en el puerto 3000
29.     aplicación escuchar ( puerto , ( ) => {
30.         consola. log ( `Servidor escuchando en el puerto
   $ { puerto } ` )
31.     } )
```

Intex.html

```
1. <!DOCTYPEhtml>
2. < html idioma = "es" >
3. < cabeza >
4.     < juego de caracteres meta = "UTF-8" >
5.     < meta http-equiv = "X-UA-
   Compatible" content = "IE=edge" >
```

```
6.      < meta name = "viewport" content = "width=device-width,
   initial-scale=1.0" >
7.      < title > Medio Distancia < / title >
8.      < secuencia
   de comandos src = "https://ajax.googleapis.com/ajax/libs/jque
   ry/3.5.1/jquery.min.js" >< / secuencia de comandos >
9.      < enlace rel = "hoja de
   estilo" href = "./Estilo/estilo.css" >
10.     < / cabeza >
11.         < cuerpo >
12.
13.             < div clase = "formLayout" >
14.                 < clase
   de etiqueta = "títulos" >       < / etiqueta >< br >< br >
15.                 <center>< label class = "títulos" > Quantum
   Tecno.com < / label >< br >< br >
16.                 < img src = "quantum.png" ancho = "200"
17.                altura = "200" >< / centro>
18.             < / div >
19.             < div clase = "formLayout" >
20.
21.                 < div clase = "formDiv" >
22.                     < br >
23.                     < label class = "titles" > Proyecto
   Final < / label >< br >< br >
24.                     < label class = "titles" > Medidor de
   Distancia Ultrasónico < / label >< br >< br >
25.                     < label class = "titles" > Detector de
   Distancias Menores a 10 cm < / label >< br >< br >
26.                 < / div >
27.
28.                 <!-- Botones -->
29.                 < br >< br >
30.                 < clase div = "contenedor de botones" >
31.                     <center>< button onclick = "toggleLed()"
    class = "button" id = "btn-
   led" > Medidor: < span >< / span > < / button >< / center >
32.                     < button onclick = "toggleMotor()" class
    = "button" id = "btn-motor" > Comprobar
   Estado: < span >< / span > < / button >
33.                 < / div >
34.                 < br >< br >< br >
35.             < / div >
36.
37.             <!-- Cargar en el archivo de configuración los
   nuevos datos-->
38.             < tipo
   de script = "texto/javascript" src = "./Scripts/setConfig.js"
    >< / script >
```

```
39.             <!-- Cargar los valores del archivo y
   sustituirlos en el formulario-->
40.             < tipo
   de script = "texto/javascript" src = "./Scripts/loadData.js"
   >< / script >
41.         < / cuerpo >
42.     < / html >
```

GitHub.

https://github.com/bboycampos01/Sensor-Ultrasonico.git