Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

"GESTIÓN DE UN PROCESO INDUSTRIAL MEDIANTE SISTEMA MICROCONTROLADOR"

Alumno: Pablo Gómez Huertas

Tutor: Prof. D. Antonio Abarca Álvarez
Dpto: Ingeniería Electrónica y Automática

Septiembre, 2018

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Ingenieria Electronica y Automática

Don Antonio Abarca Álvarez , tutor del Proyecto Fin de Carrera titulado: "Gestión de un proceso industrial mediante sistema microcontrolador", que presenta Pablo Gómez Huertas, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2018

El alumno:

Pablo Gómez Huertas

El tutor:

ABARCA ALVAREZ ANTONIO - 23786769P

Firmado digitalmente por ABARCA ALVAREZ ANTONIO - 23786769P
Nombre de reconocimiento (DN): c=ES, serialNumber=23786769P, sn=ABARCA ALVAREZ, givenName=ANTONIO, cn=ABARCA ALVAREZ ANTONIO - 23786769P
Fecha: 2018.09.05 11:33:39 +02'00'

Antonio Abarca Álvarez

3

Índice

1. MEMORIA

1.1. OBJETIVO

El objetivo de este proyecto es el diseño y realización de un proceso industrial basado en el control de una cinta transportadora controlada mediante una pantalla táctil, utilizando Arduino UNO para controlar el proceso.

1.2. ALCANCE

El alcance consiste en, mediante un motor, hacer que la cinta gire y gracias a los sensores poder tener un control del proceso. Todo el proceso se puede comprobar mediante la pantalla táctil.

1.3. ANTECEDENTES

Este proyecto ha sido realizado (a modo de maqueta) ante la necesidad de que cualquier empresa necesite optimizar su trabajo mediante una cinta transportadora.

1.4. NORMAS Y REFERENCIAS

1.4.1. BIBLIOGRAFIA

Para la realización de este proyecto, la siguiente bibliográfica ha sido de gran utilidad, a parte de algunos datasheets de componentes y manuales de algunos de los programas utilizados.

Store.arduino.cc. (2018). Arduino Nano. [Online] Disponible en: https://store.arduino.cc/arduino-nano .

Itead.cc. (2018). Nextion HMI Solution - ITEAD Wiki. [online] Disponible en: https://www.itead.cc/wiki/Nextion_HMI_Solution .

Forum.arduino.cc. (2018). Español. [online] Disponible en: https://forum.arduino.cc/index.php?board=32.0 .

Sensor de temperatura y humedad DHT11 y DHT22 [online] Disponible en: https://naylampmechatronics.com/blog/sensor-de-temperatura-y-humedad-DHT1.html

Sensor de distancia ultrasonido HC-SR04 [online] Disponible en: https://www.luisllamas.es/sensor-de-ultrasonidos-hc-sr04/

Modulo controlador de motores L298N [online] Disponible en: https://www.prometec.net/l298n/

1.4.2 SOFTWARE UTILIZADO

Los principales software utilizados para el proyecto son los siguientes:

Arduino versión 1.6.9, desarrollado por Arduino.cc. Software utilizado para la programación de placas Arduino o compatibles.

- Microsoft Office 2007, desarrollado por Microsoft. Software ofimático.
- Nextion editor versión 0.53, desarrollado por Nextion. Software utilizado para la programación de la pantalla TFT.
- Proteus Desing 8, desarrollado por Labcenter Electronics, utilizado para la simulación de la comunicación puerto serie.

1.5. CONCEPTOS A TENER EN CUENTA

1.5.1 SEÑAL PWM

En este apartado vamos a ver los fundamentos en los que se basa la generación de salidas analógicas en Arduino. El procedimiento para generar una señal analógica es el llamado PWM. Señal PWM (Pulse-width modulation) señal de modulación por ancho de pulso. Donde:

- PW (Pulse Width) o ancho de pulso, representa al ancho (en tiempo) del pulso.
- length/period (periodo), o ciclo, es el tiempo total que dura la señal.

La frecuencia se define como la cantidad de pulsos (estado on/off) por segundo y su expresión matemática es la inversa del periodo, como muestra la siguiente ecuación.

$$frequency = \frac{1}{period}$$

El periodo se mide en segundos, de este modo la unidad en la cual se mide la frecuencia (Hz) es la inversa a la unidad de tiempo (segundos).

Existe otro parámetro asociado o que define a la señal PWM, denominado "Duty cycle", Ciclo de Trabajo, el cual determina el porcentaje de tiempo que el pulso (o voltaje aplicado) está en estado activo (ON) durante un ciclo.

En la ilustración 1 se puede ver a diferentes señales PWM con distintos “Duty cycles”.

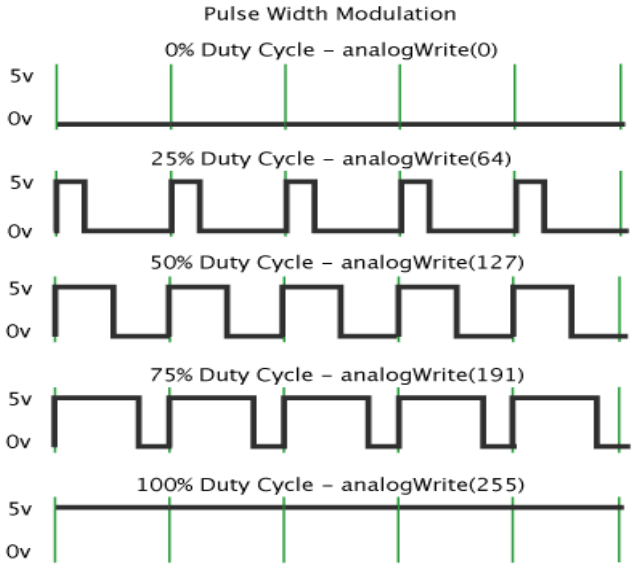


Ilustración 1: Señal PWM

La señal PWM la hemos utilizado en el proyecto para tener el control de velocidad del motor DC (si decrementas el ciclo de trabajo sobre la señal de

control del circuito de potencia que actúa sobre el motor el motor se mueve más lentamente), ajustar la intensidad de brillo de un LED, etc.

1.6. REQUISITOS DE DISEÑO

Los requisitos son poder controlar la velocidad de operación del motor, con los sensores ultrasonidos poder realizar un sistema de control de la cinta y con el sensor temperatura obtener valores de temperatura y humedad. Todos los valores que nos proporcionen los componentes se mostraran mediante una pantalla con la que el usuario también podrá interactuar con ella.

Para todo esto debemos crear un circuito de alimentación para poder trabajar con el modulo L298. Para el procesamiento de los datos haremos uso de una placa Arduino UNO.

Para el transporte de los objetos será necesaria la creación de una cinta transportadora.

1.7. COMPARATIVA DE POSIBLES SOLUCIONES

Ante los requisitos de diseño, para la interacción con el usuario del sistema, se ha optado por utilizar una pantalla TFT, de manera que en dicha pantalla se muestra los datos requeridos, de distancia, cantidad, temperatura, humedad, y el control de la velocidad del motor. De la misma manera, este tipo de pantallas permiten crear botones táctiles, que servirán de entradas/salidas para nuestro sistema, para elegir entre las diferentes opciones de visualización.

Para poder accionar la cinta transportadora, utilizaremos un motor de 12V dc. La unión del rodillo de la cinta con el motor se realizara mediante engranajes. Uno conectado a un rodillo de la cinta y otro al motor dc.

La iteración del Arduino con el motor la controlaremos con un modulo L298N, que a su vez nos proporcionara una salida de 5V para poder conectar la pantalla.

Mediante el sensor de ultrasonidos podremos contabilizar el número de paquetes que han pasado, así como poder realizar un sistema de control con el sensor y el pare del motor.

Para medir la temperatura haremos uso de un sensor DHT22 que nos proporcionará (a través de la pantalla) los valores de humedad y temperatura de donde se encuentre.

El microcontrolador utilizado es un Arduino UNO, ya que ofrece un sistema de control adecuado para el proceso que queremos realizar. Además presenta un bajo consumo de energía eléctrica y una velocidad de muestreo suficiente para poder trabajar bien con los pines que vamos a utilizar.

1.8. RESULTADOS FINALES

Como resultados finales, explicaré la descripción de cada uno de los componentes que he utilizado para poder realizar el proyecto, incluyendo la programación tanto de la pantalla TFT como del microcontrolador de Arduino.

Los componentes son:

- Sensor Ultrasonidos HC-SR04
- Sensor Temperatura DHT22
- Transformador 12V RS-PRO
- Modulo L298N
- Motor DC
- Pantalla Nextion
- Placa Arduino UNO
- Programación Arduino
- Maqueta

1.8.1. SENSOR ULTRASONIDOS HC-SR04

Para poder realizar la contabilización de objetos haremos uso del sensor ultrasonidos.

Ilustración 2: Sensor ultrasonidos

Este sensor (Ilustración 2) es capaz de medir distancia de hasta 4.5 m con una resolución de 0.3 cm. Ajustando el código en Arduino (explicado más adelante) a una distancia determinada podemos realizar la cuenta de objetos.

Las principales características del sensor son las siguientes:

- Tensión de alimentación: 5 Vcc
- Frecuencia de trabajo: 40 KHz
- Rango máximo: 4.5 m
- Rango mínimo: 1.7 cm
- Duración mínima del pulso de disparo (nivel TTL): 10 µS.
- Duración del pulso eco de salida (nivel TTL): 100-25000 µS.

Los pines de conexión del sensor son 4:

- VCC
- Trigger (*Disparo del ultrasonido*)
- Echo (*Recepción del ultrasonido*)
- GND

El funcionamiento del sensor es bastante sencillo, consiste en emitir un sonido ultrasónico por uno de sus transductores (trigger) y esperar que el

sonido rebote de algún objeto presente, el eco es captador por el segundo transductor (echo). La distancia es proporcional al tiempo que tarda en devolver la señal del eco. En la ilustración 3 se muestra el funcionamiento de modo gráfico.

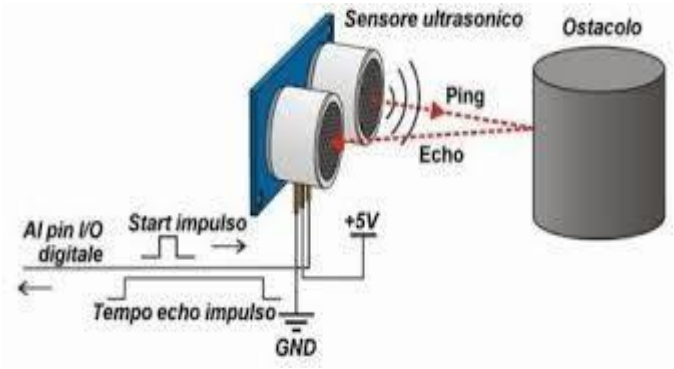


Ilustración 3: Funcionamiento sensor ultrasonidos

1.8.2. SENSOR TEMPERATURA DHT22

Con este sensor (Ilustración 4) conseguiremos las medidas de humedad y temperatura que reflejaremos en la pantalla.

Ilustración 4: Sensor DHT22

Este tipo de sensor dispone de un procesador interno que realiza el proceso de medición, proporcionando la medición mediante una señal digital, por lo que resulta muy sencillo obtener la medición desde un microprocesador como Arduino.

Las características del DHT22 son las siguientes:

- Alimentación: 3.3v – 5.5v, tomando como valor recomendado 5v.
- Medición de temperatura entre -40 a 125, con una precisión de 0.5ºC.
- Medición de humedad entre 0 a 100%, con precisión del 2-5%.
- Frecuencia de muestreo de 2 muestras por segundo (2 Hz).

El montaje del sensor con el Arduino es bastante sencillo ya que disponemos de 4 patillas, de las cuales usaremos 3, Vcc, Output y GND, como se muestra en la ilustración 5.

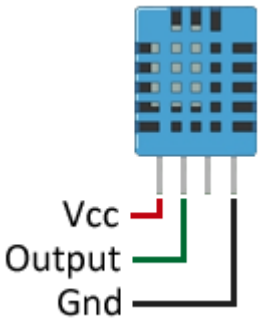


Ilustración 5: Patillaje del sensor DHT22

Conectar el sensor es sencillo, simplemente alimentamos desde Arduino al sensor a través de los pines GND y Vcc del mismo. Por otro lado, conectamos la salida Output a una entrada digital de Arduino. El sensor que nosotros hemos utilizado ya viene soldado a una placa por lo que no es necesario utilizar una resistencia entre Vcc y output.

1.8.3. TRANSFORMADOR 12V RS-PRO

Nuestro proyecto se conectará a la red eléctrica que suministra 230V AC eficaces y 50Hz. Dado que para controlar el motor vamos a utilizar un modulo L298N que se alimenta con 12V, necesitaremos hacer uso de un transformador de 230V a 12V para alimentar el modulo. En el caso de la placa Arduino UNO haremos uso del mismo.

El transformador que hemos utilizado es el que se muestra en la figura 6 siguiente:

Ilustración 6: Transformador

Las principales características de este trasformador son las siguientes:

- Tensión entrada: 230V ac
- Tensión salida: 12V dc
- Potencia nominal: 12 W
- Número de salidas: 1
- Corriente de salida: 1 A
- Tipo de conversión de la señal de alimentación: ac-dc
- Frecuencia de funcionamiento entre 47Hz y 63 Hz
- Temp. máx. ambiente +40 °C

1.8.4. MODULO L298N

Este módulo basado en el chip L298N te permite controlar dos motores de corriente continua o un motor paso a paso bipolar de hasta 2 amperios.

Ilustración 7: Modulo L298N

El módulo cuenta con todos los componentes necesarios para funcionar sin necesidad de elementos adicionales, entre ellos diodos de protección y un regulador LM7805 que suministra 5V a la parte lógica del integrado L298N.

Cuenta con jumpers de selección para habilitar cada una de las salidas del módulo (A y B). La salida A esta conformada por OUT1 y OUT2 y la salida B por OUT3 y OUT4. Los pines de habilitación son ENA y ENB respectivamente. En la siguiente imagen (ilustración 8) se muestra como se distribuye las conexiones en la placa del modulo:

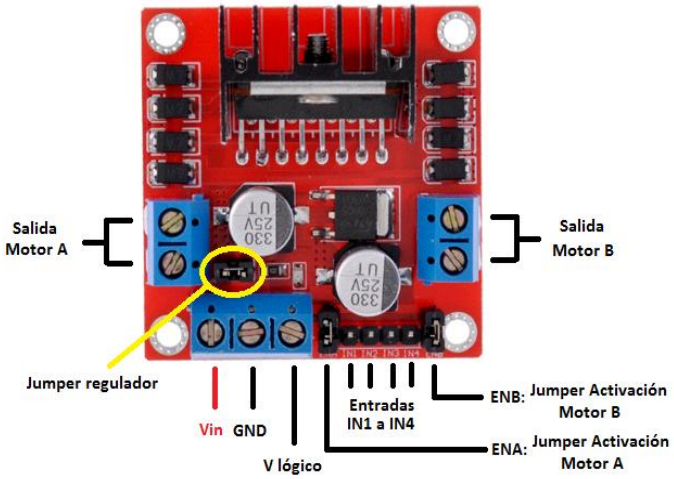


Ilustración 8: Esquema modulo L298N

Este módulo se puede alimentar de 2 maneras gracias al regulador integrado LM7805.

El primero (y el que hemos utilizado para nuestro proyecto) se da cuando el jumper de selección de 5V se encuentra activo, haciendo que el módulo permita una alimentación de entre 6V a 12V DC**.** Como el regulador se encuentra activo, el pin marcado como +5V tendrá un voltaje de 5V DC. Este voltaje se puede usar para alimentar la parte de control del módulo ya sea un microcontrolador o un Arduino (en nuestro caso para alimentar la pantalla TFT), pero recomendamos que el consumo no sea mayor a 500 mA.

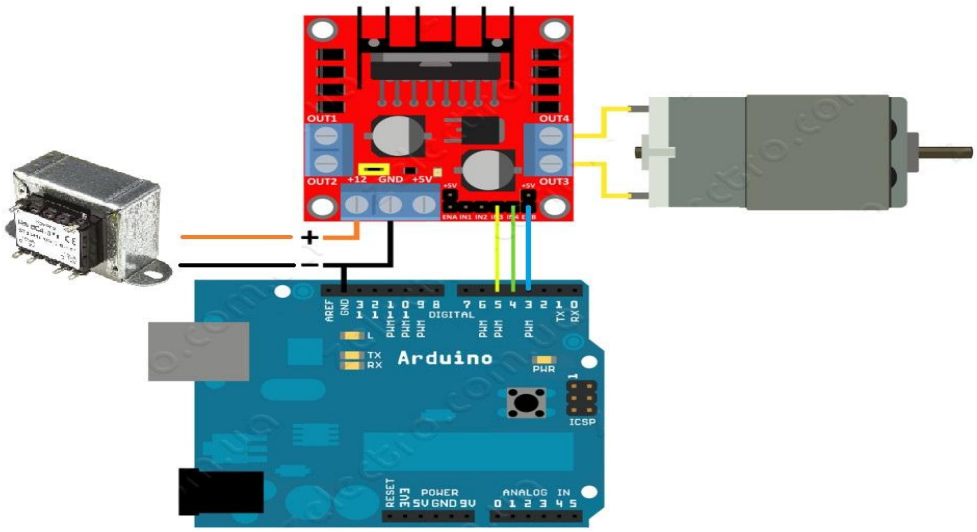


Ilustración 9: Esquema montaje modulo L298N

Este segundo es cuando el jumper de selección de 5V se encuentra inactivo, el módulo permite una alimentación de entre 12V a 35V DC. Como el regulador no está funcionando, tendremos que conectar el pin de +5V a una tensión de 5V para alimentar la parte lógica del L298N.

1.8.5 MOTOR DC

Para poder hacer girar la cinta transportadora haremos uso de un motor dc de 12V. El motor se conectara mediante dos cables a los pines de salida del modulo L298N (salida B). En la siguiente imagen se muestra el motor que hemos utilizado.

Ilustración 10: Motor dc 12V

En el eje del motor colocaremos un engranaje que se conectará a uno de los rodillos de la cinta y será el encargado de hacer que ande la cinta. Las características de este motor son las siguientes:

- Rango de voltaje: 1,5 – 12V
- Longitud del eje del motor: 10mm
- Velocidad nominal a 3V: 50 RPM/min
- Velocidad nominal a 6V: 100 RPM/min
- Velocidad nominal a 12V: 200 RPM/min

1.8.6 PANTALLA NEXTION

Para el circuito referente a la pantalla, necesitamos alimentar ésta a 5V (lo realizaremos con el modulo L298N), y las conexiones al microcontrolador mediante comunicación por puerto serie, por lo que tenemos que conectar los pines RX y TX de la pantalla y el microcontrolador. Para ello conectamos el pin RX del microcontrolador al TX de la pantalla, y el pin TX del microcontrolador al RX de la pantalla.

La pantalla que hemos utilizado es la Nextion NX4832T035_011 HMI TFT LCD, una pantalla táctil de tipo resistiva, en la Ilustración 11 se puede ver la pantalla, y posteriormente sus principales características.

Ilustración 11: Pantalla Nextion

Las principales características son:

- Resolución de pantalla 480x320
- RGB 65K
- Pantalla TFT con panel táctil resistivo de 4 hilos integrado
- Ranura micro-SD para actualización de firmware
- Memoria Flash 16M
- Fuente recomendada DC 5V 500mA
- Consumo energético 5V 145mA (0.725W)

Para la programación de la pantalla haremos uso del software que nos proporciona el fabricante de este tipo de pantalla. Se trata de una programación gráfica, de acuerdo a cuando estamos diseñando la interfaz con el usuario, aunque también podemos añadir algunos comandos para determinadas acciones como se verá más adelante.

Para empezar a crear un proyecto en Nextion es básicamente como siempre; vas a menú principal, “File” – “New”, guardamos en el directorio que queremos trabajar. Después de esto se abrirá una ventana y diferente a otros programas, es el de elegir el tipo de pantalla que tenemos, como se ve en la figura 12.

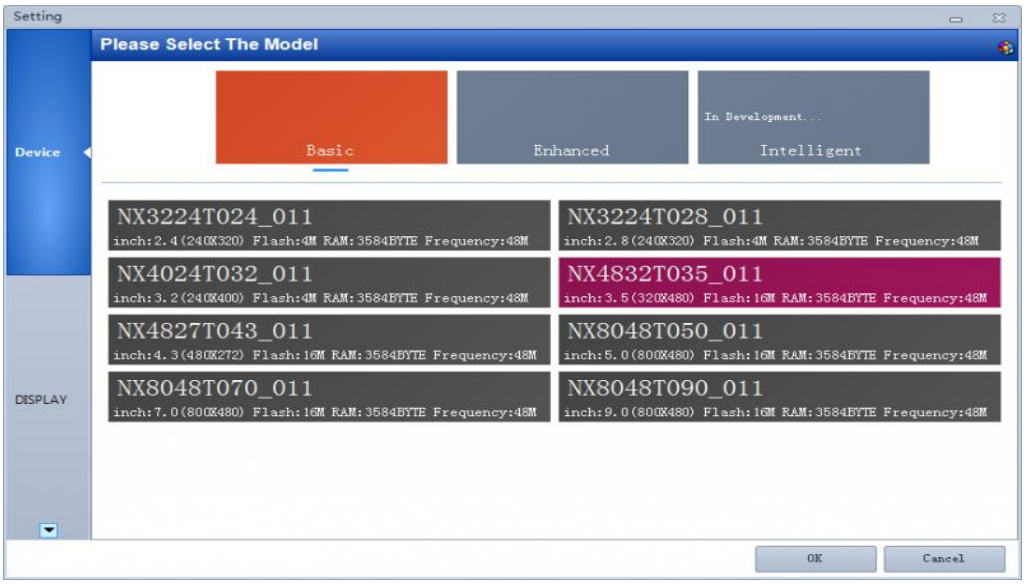


Ilustración 12: Menú para escoger nuestra pantalla

Ahora comenzaré a explicar las ventanas más importantes del programa que hemos utilizado (Nextion Editor).

- Toolbox

En esta ventana aparecen todos los elementos que podemos utilizar para la interfaz como son los recuadros de texto, botones, slider y otros más. En la imagen 13 se pueden ver todas estas herramientas, de las cuales sólo se van a explicar las utilizadas en este proyecto.

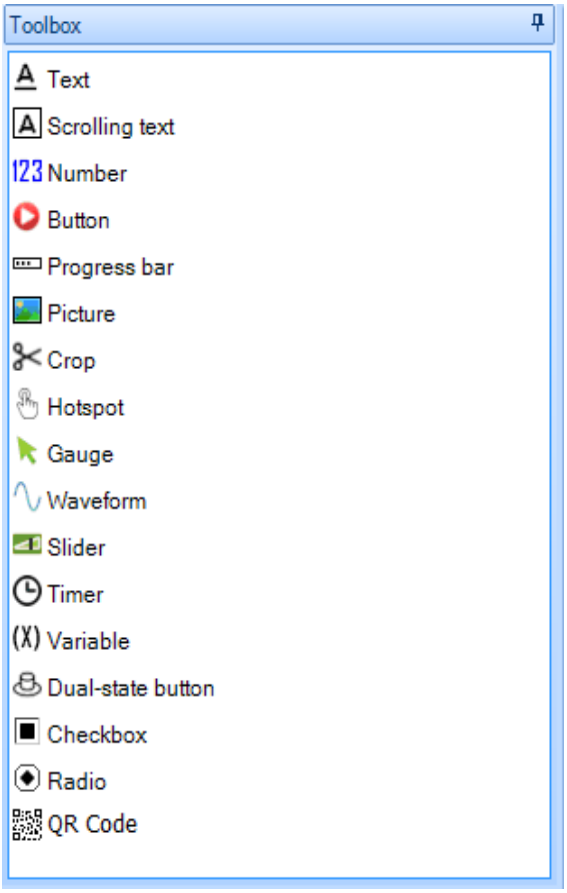


Ilustración 13: Panel Toolbox Nextion

De esta ventana solo hemos utilizado tres comandos:

- Text: para poner notas de texto en la pantalla.
- Number: para poner recuadros numéricos donde mostrar los datos obtenidos desde Arduino.
- Button: utilizados para movernos entre páginas y comunicarnos con Arduino.

- Pictures

En esta ventana se cargan al programa las imágenes que utilizaremos de una forma u otra en nuestra interfaz.

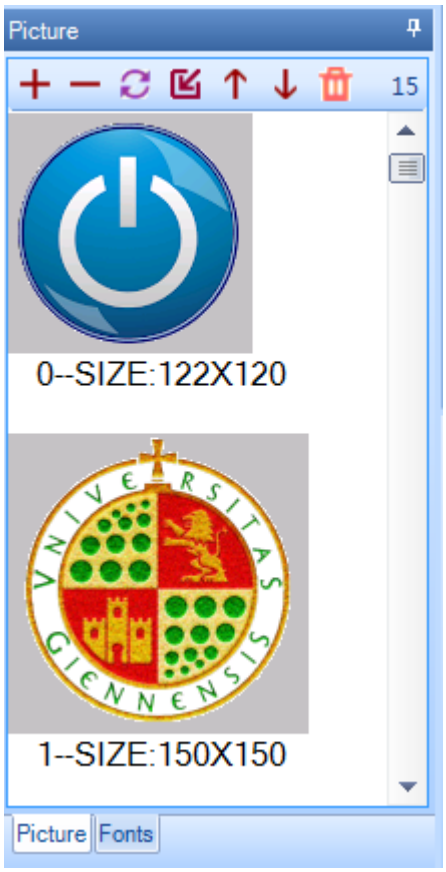


Ilustración 14: Panel Picture Nextion

Para añadir las imágenes hay que pinchar sobre el símbolo de suma, y buscar las imágenes en la carpeta que las tengamos guardadas e introducirlas para utilizar en nuestra interfaz. En la Ilustración 14 se pueden ver algunas de las imágenes utilizadas para este proyecto, cuyo pie de imagen es el número de imagen y el tamaño en píxeles que contiene.

- Page

Aquí aparecerán todas las páginas que tiene nuestro proyecto.

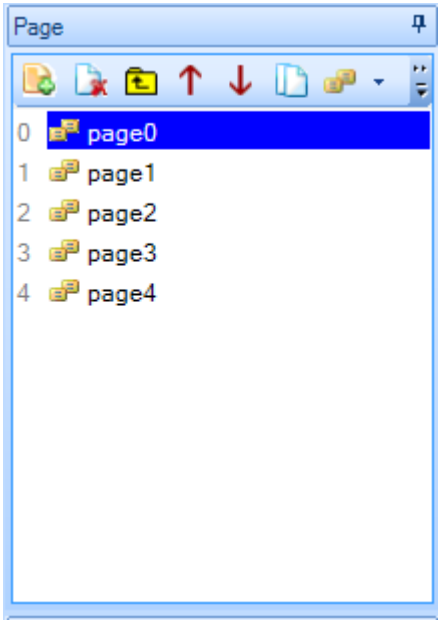


Ilustración 15: Panel Page Nextion

De esta ventana cabe destacar que para añadir una nueva página hay que pinchar sobre el primer elemento (hoja de color amarillo con el signo de suma verde), y para eliminar una imagen pinchar justo el de al lado (hoja blanca con una cruz roja). Aparecerá recuadrada de azul la página en la que estemos en cada momento, y pinchando en cada página accederemos a dicha página para su programación.

- Attribute

En esta ventana se mostrará toda la información que contiene cada elemento.

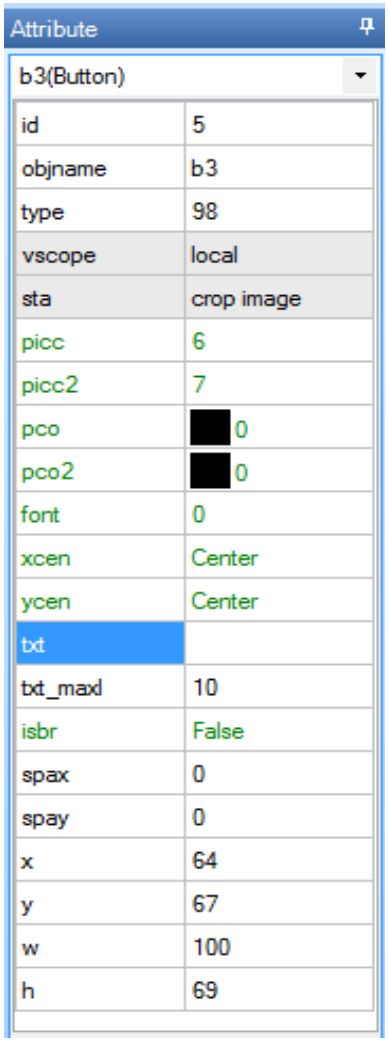


Ilustración 16: Panel Attribute Nextion

Aquí se puede ver toda la información de botón b3, ahora explicaré que función realiza cada menú que hemos utilizado en el proyecto.

- Id: corresponde con el identificador de este elemento.
- Objname: indica el nombre de dicho elemento (le puedes dar el que quieras)
- Sta: utilizado para el tipo de fondo que tendrá este objeto, del cual hay tres posibilidades. Solid color, el elemento tendrá un fondo de un solo color. Image, tendrá de fondo una imagen predeterminada, y se adaptara el objeto al tamaño de dicha imagen. Crop image, similar al anterior, pero este elemento no tiene por qué tener el mismo tamaño que la imagen
- Picc y picc2: indica el número de la imagen q tienen de fondo.
- Pco y pco2: indica el color de letra del texto del elemento.

- Txt: para poner una nota de texto en el elemento.
- X, Y: son las coordenadas del vértice superior izquierdo del elemento dentro de la página.
- W, H: indica el tamaño de dicho elemento.

- Event

En esta ventana podemos declarar algunos comandos, principalmente en botones, para que realicen una función determinada.

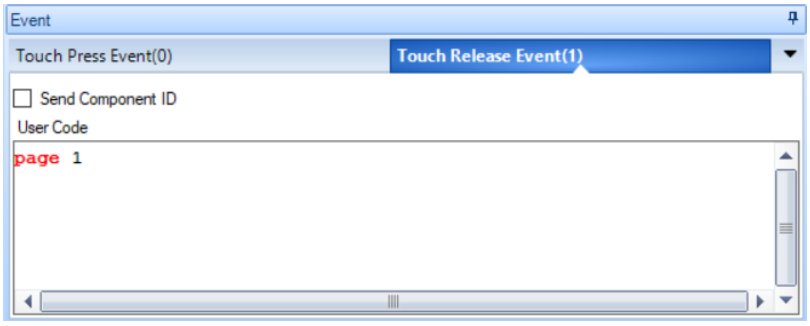


Ilustración 17: Panel Event Nextion

Las principales funciones de esta ventana son:

- Send Component ID: si se habilita esta opción, se enviara dicha información al Arduino cuando se pulse sobre el botón.
- User code: donde podemos escribir las líneas de código que debe realizar la pantalla por si misma sin necesidad de comunicación con Arduino. En el proyecto solo hemos utilizado el comando “page () “ donde () es el numero de pagina que queremos que se desplace el botón
- Touch Press Event: donde se puede habilitar las anteriores opciones. Estas opciones las hará cuando se pulse sobre el botón.
- Touch Release Event: donde se puede habilitar las anteriores opciones. Estas opciones las hará cuando se suéltela pulsación sobre el botón.

Ahora explicare las páginas que contiene nuestra pantalla Nextion, explicando así cada elemento que contenga.

- Page 0

Ilustración 18: Page 0 pantalla Nextion

- Página de inicio page 0, con id 0, con color solido de fondo
- Botón b0 que nos llevará a la página 1 y la que hace las veces de menú, id 1, sta: image, picc y picc2: 0, pco:0 (color negro), pco2: 63488 (color negro), x: 284, y: 149, w: 122, h: 120. Touch Release Event: User code (page 1).
- Texto t0, id 2, sta: solid color, style: flat, pco: 0 (color blanco), txt: START TFG, x: 277, y: 67, w: 130, h: 62.
- Texto t1, id 4, sta: solid color, style: flat, pco: 0 (color negro), txt: PABLO GOMEZ HUERTAS, x: 19, y: 254, w: 180, h: 30.
- Texto t2, id 5, sta: solid color, style: flat, pco: 0 (color negro), txt: UNIVERSIDAD DE JAEN, x: 25, y: 190, w: 100, h: 56.
- Picture p0, id 3, vscope: local, pic 1, x: 9, y: 37, w: 150, y: 150.

- Page 1

Ilustración 19: Page 1 pantalla Nextion

- Página de inicio page 1, con id 0, con imagen de fondo pic 3.
- Botón b0 que nos llevará a la página 2, donde accederemos al menú del motor, id 1, sta: solid color, bco: color gris, bco2: color verde, pco: 0 (color negro), pco2: 63488 (color negro), x: 160, y: 45, w: 159, h: 51. Touch Release Event: User code (page 2).
- Botón b1 que nos llevará a la página 3, donde accederemos al menú del sensor de distancia, id 2, sta: solid color, bco: color gris, bco2: color verde, pco: 0 (color negro), pco2: 63488 (color negro), x: 160, y: 123, w: 159, h: 51. Touch Release Event: User code (page 3).
- Botón b2 que nos llevará a la página 4, donde accederemos al menú de la temperatura, id 3, sta: solid color, bco: color gris, bco2: color verde, pco: 0 (color negro), pco2: 63488 (color negro), x: 160, y: 204, w: 159, h: 51. Touch Release Event: User code (page 4).

- Page 2

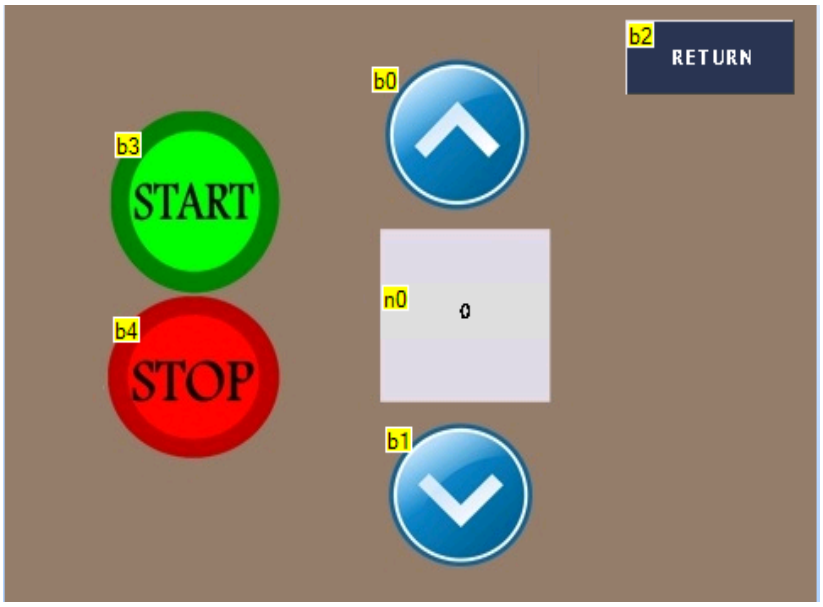


Ilustración 20: Page 2 pantalla Nextion

- Página de menú del motor page 2, con id 0, con imagen de fondo pic 6.
- Botón b0 aumentara la velocidad del motor, id 2, sta: crop image, picc: 6, picc2: 7, pco: 0 (color negro), pco2: 63488 (color negro), x: 216, y: 32, w: 100, h: 72. Touch Release Event: Send Component ID.
- Botón b3 ajustara directamente la velocidad del motor a una implementada en código, id 5, sta: crop image, picc: 6, picc2: 7, pco: 0 (color negro), pco2: 63488 (color negro), x: 64, y: 67, w: 100, h: 70. Touch Release Event: Send Component ID.
- Botón b1 disminuirá la velocidad del motor, id 3, sta: crop image, picc: 6, picc2: 7, pco: 0 (color negro), pco2: 63488 (color negro), x: 224, y: 225, w: 100, h: 70. Touch Release Event: Send Component ID.
- Botón b4 parara el motor directamente, id 6, sta: crop image, picc: 6, picc2: 7, pco: 0 (color negro), pco2: 63488 (color negro), x: 63, y: 166, w: 100, h: 70. Touch Release Event: Send Component ID.
- Botón b2 para volver a la página 1 de menús, id 4, sta: solid color, bco: color azul, bco2: color verde, pco: 0 (color blanco), pco2:

65535 (color blanco), txt: RETURN, x: 367, y: 8, w: 100, h: 40. Touch Release Event: User code (page 1).

- Número n0 donde visualizaremos el valor de velocidad del motor procedente desde Arduino, id 1, sta: solid color, bco: 57083(color gris), pco: 0 (color negro), x: 222, y: 149, w: 100, h: 30.

- Page 3

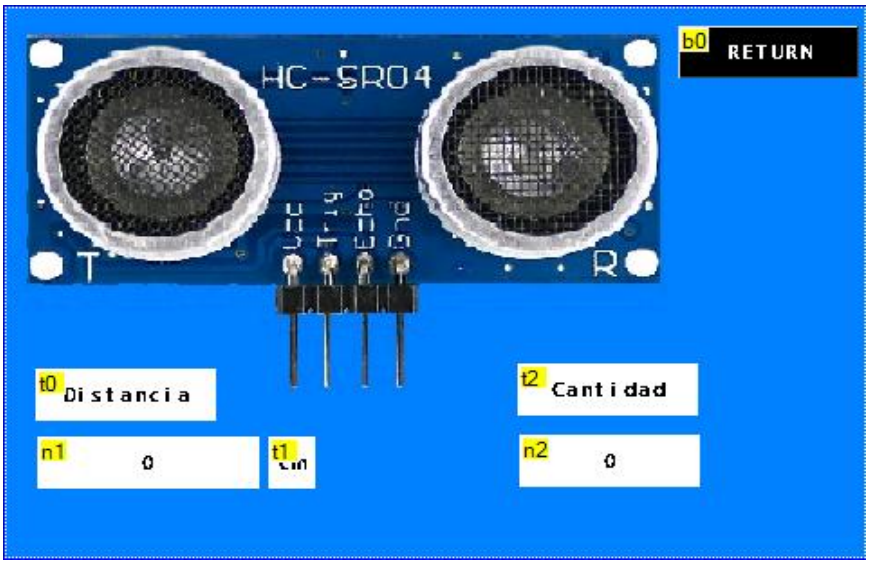


Ilustración 21: Page 3 pantalla Nextion

- Página de menú del sensor de distancia page 3, con id 0, con imagen de fondo pic 8.
- Botón b0 para volver a la página 1 de menús, id 1, sta: solid color, bco: color negro, bco2: color verde, pco: 0 (color blanco), pco2: 65535 (color blanco), txt: RETURN, x: 375, y: 13, w: 100, h: 30. Touch Release Event: User code (page 1).
- Número n1 donde visualizaremos el valor de la distancia que medirá el sensor, id 2, sta: solid color, bco: 65535(color blanco), pco: 0 (color negro), x: 21, y: 249, w: 123, h: 30. Touch Release Event: Send Component ID.
- Número n2 donde visualizaremos la cantidad de objetos que pasan por la cinta, id 6, sta: solid color, bco: 65535(color blanco), pco: 0 (color negro), x: 287, y: 248, w: 100, h: 30. Touch Release Event: Send Component ID.

- Texto t0, id 3, sta: solid color, style: flat, bco: 65535 (color blanco), txt: Distancia, x: 19, y: 210, w: 100, h: 30.
- Texto t1, id 4, sta: solid color, style: flat, bco: 65535 (color blanco), txt: cm, x: 148, y: 249, w: 26, h: 30.
- Texto t2, id 5, sta: solid color, style: flat, bco: 65535 (color blanco), txt: Cantidad, x: 286, y: 207, w: 100, h: 30.

- Page 4

Ilustración 22: Page 4 pantalla Nextion

- Página de menú del sensor de temperatura page 4, con id 0, con imagen de fondo pic 5.
- Botón b0 para volver a la página 1 de menús, id 4, sta: solid color, bco: color negro, bco2: color verde, pco: 0 (color blanco), pco2: 65535 (color blanco), txt: RETURN, x: 359, y: 17, w: 100, h: 30. Touch Release Event: User code (page 1).
- Texto t0, id 2, sta: crop image, picc: 5, pco: 65535 (color blanco), Font: 1, txt: %, x: 241, y: 223, w: 53, h: 42.
- Texto t2, id 1, sta: crop image, picc: 5, pco: 65535 (color blanco), Font: 1, txt: Humedad, x: 30, y: 216, w: 98, h: 57.
- Texto t3, id 3, sta: crop image, picc: 5, pco: 65535 (color blanco), Font: 4, txt: º, x: 249, y: 41, w: 44, h: 45.

- Número num0 donde visualizaremos el valor de temperatura obtenido por el sensor DHT22, id 7, sta: crop image, picc: 5, pco: 65535 (color blanco), Font: 3, x: 93, y: 56, w: 145, h: 147. Touch Release Event: Send Component ID.
- Número num1 donde visualizaremos el valor de humedad obtenido por el sensor DHT22, id 8, sta: crop image, picc: 5, pco: 65535 (color blanco), Font: 2, x: 133, y: 229, w: 100, h: 30. Touch Release Event: Send Component ID.

1.8.7 PLACA ARDUINO UNO

En cuanto al microcontrolador, es el encargado de gestionar la información, realizar los cálculos oportunos y enviarlos a la pantalla. Para ello hemos hecho uso de Arduino UNO, ya que tiene un bajo consumo y es sencillo de utilizar. En la figura 23 se muestra como es físicamente la placa Arduino UNO que hemos utilizado.

Ilustración 23: Placa Arduino UNO

La placa puede alimentarse a través del cable USB o mediante una fuente de alimentación externa (siempre comprendida entre un rango de 7-12V), como

puede ser un pequeño transformador o directamente con el conector Jack de 2,1mm con el positivo en el centro.

Otra manera de poder alimentarlo es con los pines Vin y GND marcados sobre la placa.

En cuanto a memoria, tenemos que el ATmega328 tiene una memoria flash de 32 KB, (0,5 KB ocupados por el Bootloader). El ATmega328 tiene 2 KB de SRAM y 1 KB de EEPROM.

En la Ilustración 24 se puede ver la información de cada patilla de este dispositivo.

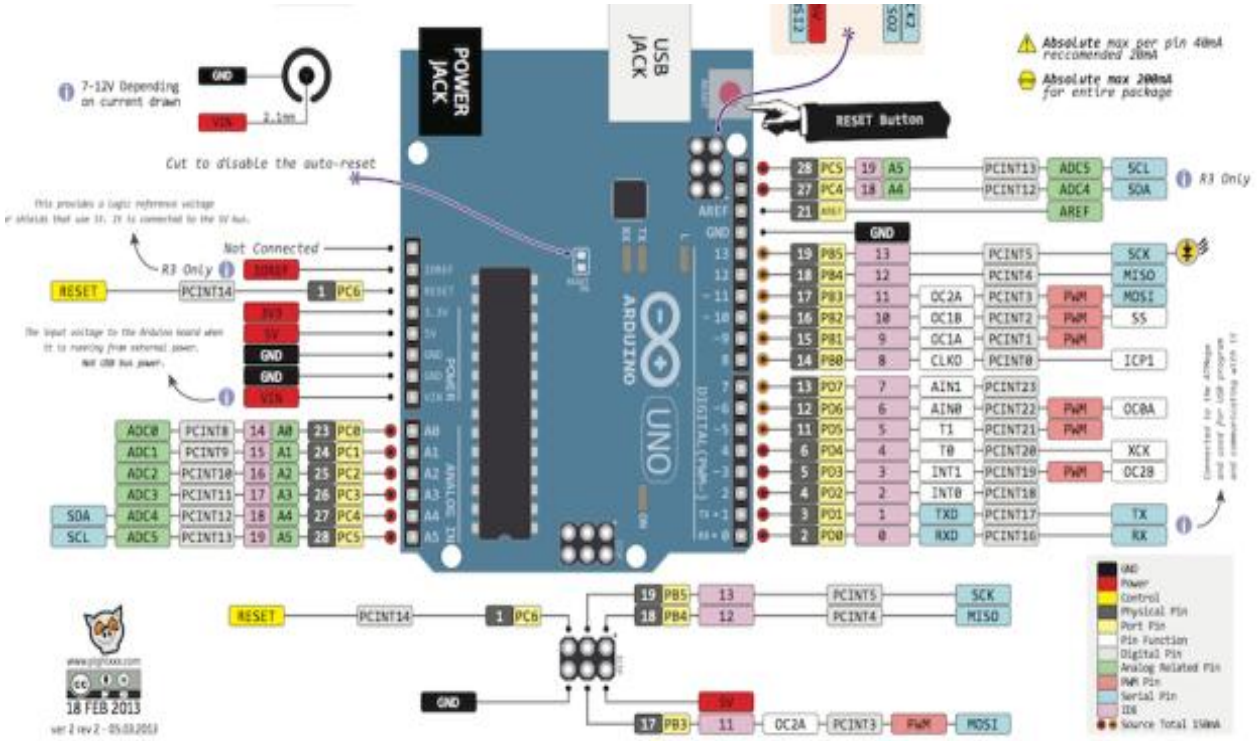


Ilustración 24: Diagrama Pinout placa Arduino UNO

Un Arduino dispone de 14 pines que pueden configurarse como entrada o salida y a los que puede conectarse cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de 0 y 5V. Las funciones que mas hemos usado para estos pines son pinMode (), digitalWrite () y digitalRead (). Cada pin puede suministrar hasta 40 mA. La intensidad máxima de entrada también es

de 40 mA. Cada uno de los pines digitales dispone de una resistencia pull-up interna de entre 20 KΩ y 50 KΩ que viene desconectada por defecto, salvo que indiquemos lo contrario.

Algunos pines especiales de Arduino son:

- Serie: 0 (RX) y 1 (TX). Se usa para recibir (RX) y transmitir (TX) datos en serie TTL.
- Interrupciones externas: 2 y 3. Estas clavijas se pueden configurar para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor. Para ello se utiliza la función attachInterrupt ().
- PWM: 3, 5, 6, 9, 10 y 11. Proporcionan salida PWM de 8 bits con la función analogWrite ().
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines admiten la comunicación SPI, que, aunque proporcionada por el hardware subyacente, no se incluye actualmente en el lenguaje Arduino.
- I^2C: Permite establecer comunicaciones a través de un bus I^2C.

Arduino también dispone de 6 pines de entrada analógicos que trasladan las señales a un conversor analógico/digital de 10 bits.

Por último, en cuanto a comunicación tenemos que el Arduino UNO tiene varias facilidades para comunicarse con un ordenador, otro Arduino u otros microcontroladores. El ATmega328 proporciona comunicación serial UART TTL (5V), que está disponible en los pines digitales 0 (RX) y 1 (TX).

Un FTDI FT232RL en la placa canaliza esta comunicación en serie a través de USB y los controladores FTDI (incluidos con el software Arduino) proporcionan un puerto virtual para el software en la computadora. El software Arduino incluye un monitor serie que permite el envío de datos textuales simples hacia y desde la placa Arduino.

Los LED RX y TX de la placa parpadearán cuando los datos se transmitan a través del chip FTDI y la conexión USB a la computadora (pero no para la comunicación serial en los pines 0 y 1). Una biblioteca de SoftwareSerial permite la comunicación serial en cualquiera de los pines digitales de la placa UNO.

Para nuestro proyecto, utilizaremos el conector jack de 2,1mm para alimentar la placa Arduino. Los pines digitales para conectar tanto como ambos sensores como el modulo L298N. Por último, los pines de comunicación TTL, que son los pines digital 1 (TX) y digital 0 (RX) para la comunicación con la pantalla Nextion TFT.

1.8.8 PROGRAMACION ARDUINO

En este apartado vamos a proceder a la programación de Arduino. El lenguaje utilizado para la programación es el implementado por Arduino, basado en C, y que se programa a través de su programa Arduino.

Lo primero que se realiza en el código es declarar las librerías que utilizaremos. Estas se declararan al principio del programa como se puede ver en la siguiente imagen:

```
1 #include <Nextion.h>
2 #include "DHT.h"
```

Ilustración 25: Librerías código

La librería “Nextion.h” se utilizará para la comunicación con la panalla del fabricante Nextion, donde mostraremos los resultados. Esta librería se encarga de parte de comunicación puerto serie entre Arduino y Nextion, lo cual dicha parte no tenemos que programarla nosotros. Con la librería “DHT.h” podemos realizar fácilmente la lectura de ambos sensores y no preocuparnos por el protocolo de comunicación entre Arduino y los sensores.

A continuación declararemos las constantes mediante “#define”. Ambas están asociadas solo al sensor de temperatura.

```
5 #define DHTPIN 13                                   // Pin al que se conecta la salida del arduino.
6 #define DHTTYPE DHT22                               // Tipo de sensor que vamos usar (DHT 22).
```

Ilustración 26: Constantes mediante define

Ahora nos dispondremos a declarar las variables globales utilizadas en el programa.

```
8  ////CONSTANTES GLOBALES PARA EL CONTROL MOTOR///////////////
9  int IN3 = 5;     // Input3 conectada al pin 5 de Arduino
10 int IN4 = 4;     // Input4 conectada al pin 4 de Arduino
11 int ENB = 3;     // ENB conectada al pin 3 de Arduino
12 int flag_vel = 0; // flag de velocidad del motor
13 int vel_pwm = 0;  // flag para el control de pwm
14 ////////////////////////////////////////////////////////////////////
15 ////CONSTANTES GLOBALES SENSOR DISTANCIA//////////////////////////
16 const int Trigger = 9 ;   //Pin digital 7 para el Trigger del sensor
17 const int Echo = 8;   //Pin digital 8 para el Echo del sensor
18 unsigned long x=0; // variable global para la distancia
19 unsigned long cont=0;// variable para el contador de objetos
20 ////////////////////////////////////////////////////////////////////
21 ////CONSTANTES GLOBALES TEMPERATURA//////////////////////////////
22 float h=0; //variable para la humedad
23 float t=0; // variable para la temperatura
24 int intervaloMedidas = 1000;// variables para la actualizacion
25 int auxMillis=0;            // de la temperatura y humedad
26 ////////////////////////////////////////////////////////////////////
```

Ilustración 27: Variables globales

Seguidamente, procederemos a definir parámetros requeridos por la pantalla Nextion. Para ellos nos ayudaremos de la librería ya incluida de Nextion. Para ello definiremos los botones utilizados, así como los display numéricos y las páginas utilizadas de la siguiente forma.

- Definición de los botones: tenemos que declarar todos los botones que, de una forma u otra, tengan que comunicarse con el Arduino. Para ello utilizaremos la siguiente expresión:

NexButton nombre en Arduino=NexButton (nº de página, id, nombre en Nextion)

```
28 //definicion de los botones (pagina,id,nombre)//
29 NexButton b0 = NexButton(2, 2, "b0");
30 NexButton b1 = NexButton(2, 3, "b1");
31 NexButton b3 = NexButton(2, 5, "b3");
32 NexButton b4 = NexButton(2, 6, "b4");
33 ////////////////////////////////////////////////
```

Ilustración 28: Definición Botones Nextion

Así quedan definidos todos los botones de la pantalla que intervienen con nuestra placa Arduino.

- Definición de los apartados numéricos: para enviar desde Arduino los datos numéricos a la pantalla Nextion, hay de definirlos de la siguiente manera:

NexNumber nombre en Arduino=NexNumber (nº de página, id, nombre en Nextion)

```
36 //definicion de los numeros (pagina,id,nombre)//
37 NexNumber n0 = NexNumber(2,1,"n0");
38 NexNumber n1 = NexNumber(3,2,"n1");
39 NexNumber num0 = NexNumber(4,7,"num0");
40 NexNumber num1 = NexNumber(4,8,"num1");
41 ////////////////////////////////////////////////
```

Ilustración 29: Definición displays Nextion

A continuación definiremos la función encargado del estado de los botones.

```
43 //Funcion para consultar el estado de los botones//
44 NexTouch *nex_listen_list[] =
45 {
46   &b0,
47   &b1,
48   &b3,
49   &b4,
50   &n0,
51   &n1,
52   &num0,
53   &num1,
54   NULL
55 };
56 ////////////////////////////////////////////////////
```

Ilustración 30: Función estado botones

Ahora procederemos a definir todas las funciones que hemos creado para realizar el proyecto. Todas se llamarán en el programa principal. Se explicará una a una la función que realizan.

```
void pulsador_mas(void *ptr) // funcion para elevar la velocidad del motor//
{
  if(vel_pwm > 255){
    vel_pwm=255;
  }else{
      flag_vel=flag_vel+5;
      vel_pwm = flag_vel;
  }
  Serial.print("n0.val=");
  Serial.print(vel_pwm);
  Funcion_ff();
  if(vel_pwm <= 255 && vel_pwm >= 0){
      digitalWrite(IN3,HIGH);
      digitalWrite(IN4,LOW);
      analogWrite(ENB,vel_pwm);
  }
}
```

Ilustración 31: Función pulsador mas

- Función pulsador_mas: esta función se llamara cuando en la página 2 se pulse el botón azul con la flecha hacia arriba. Este hará que incremente la velocidad del motor de 5 en 5 y se mostrara en el display de esa página.

```
void pulsador_menos(void *ptr)//funcion para reducir la velocidad//
{
  if(flag_vel <= 0){
    vel_pwm=0;
  }else{
      flag_vel=flag_vel-5;
      vel_pwm = flag_vel;
  }
  Serial.print("n0.val=");
  Serial.print(vel_pwm);
  Funcion_ff();
  if(vel_pwm >= 0 && vel_pwm <=255){
      digitalWrite(IN3,HIGH);
      digitalWrite(IN4,LOW);
      analogWrite(ENB,vel_pwm);
  }
}
```

Ilustración 32: Función pulsador menos

- Función pulsador menos: esta función se llamara cuando en la página 2 se pulse el botón azul con la flecha hacia abajo. Este hará que se reduzca la velocidad del motor de 5 en 5 y se mostrara en el display de esa página.

```
void pulsador_stop(void *ptr) //Para inmediatamente el motor//
{
  flag_vel=0;
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,0);
  Serial.print("n0.val=");
  Serial.print(flag_vel);
  Funcion_ff();
}
```

Ilustración 33: Función pulsador stop

- Función pulsador_stop: esta función se llamara cuando en la página 2 se pulse el botón rojo de “stop”. Cuando pulsemos, el motor debe parar inmediatamente independientemente de la velocidad que lleve.

```
void pulsador_start(void *ptr) //Arranca el motor a una velocidad de 50 rpm//
{
  flag_vel=80;
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,flag_vel);
  Serial.print("n0.val=");
  Serial.print(flag_vel);
  Funcion_ff();
}
```

Ilustración 34: Función pulsador start

- Función pulsador_start: esta función se llamara cuando en la página 2 se pulse el botón rojo de “start”. La función que debe realizar es la de realizar un “arranque rápido” a una velocidad establecida ya en el código del programa a 50 rpm.

```
void Funcion_ff(void)// Función utilizada para la comunicación
{
Serial.write(0xff);
Serial.write(0xff);
Serial.write(0xff);
}
```

Ilustración 35: Función "ff"

- Función Funcion_ff: esta función es utilizada a la hora de enviar datos a la pantalla Nextion, ya que tras enviar cada dato, hay q enviar tres valores 0xff, por cómo está definido el protocolo de comunicación.

```
//funcion medida de la distancia//
unsigned long funcion_distancia(){

  long t; //timepo que tarda en llegar el eco
  long d; //distancia en centimetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
  d = t/59;              //escalamos el tiempo a una distancia en cm

  return d;

}
```

Ilustración 36: Función distancia

- Función función_distancia: esta función es la encargada de obtener los valores del sensor HC-SR04 y transformarlos a centímetros para así mostrarlos en la página 3 de la pantalla.

```
void imprimeNextion () {
  int auxhumedad = int(h);
  int temperatura = int(t);
  Serial.print("num0.val=");
  Serial.print(temperatura);
  Funcion_ff();
  String humedad = String (auxhumedad);
  Serial.print("num1.val=");
  Serial.print(humedad);
  Funcion_ff();
}
```

Ilustración 37: Función imprimeNextion

- Función imprimeNextion: con esta última función llamaremos a los valores de temperatura y humedad y los mandaremos por puerto serie a la pantalla. Estos se verán reflejados en la página 4 de la pantalla de Nextion.

Ahora que ya tenemos todas las funciones definidas, vamos a pasar a programar el setup de Arduino. La principal tarea de esta función es la inicialización de comunicaciones e interrupciones. En la imagen 38 se puede ver la programación utilizada y posteriormente viene explicado cada comando utilizado.

```
void setup()
{

 Serial.begin(9600);
 nexInit();

 dht.begin();      // Iniciamos sensor.
 imprimeNextion(); //mandamos valores iniciales

 b0.attachPop(pulsador_mas);
 b1.attachPop(pulsador_menos);
 b3.attachPop(pulsador_start);
 b4.attachPop(pulsador_stop);

 pinMode (ENB, OUTPUT); //pin como salida
 pinMode (IN3, OUTPUT); //pin como salida
 pinMode (IN4, OUTPUT); //pin como salida

 pinMode(Trigger, OUTPUT); //pin como salida
 pinMode(Echo, INPUT);  //pin como entrada
 digitalWrite(Trigger, LOW);//Inicializamos el pin con 0

}
```

Ilustración 37: Setup de Arduino

- Serial.begin (*velocidad*): este comando inicializa la comunicación serial con la pantalla a una velocidad de 9600 baudios por segundo.
- nexInit (): Inicializa la comunicación con Nextion mediante su librería.
- dht.begin (): inicializa la comunicación con el sensor de temperatura.

- imprimeNextion (): llamamos a la función para que de unos valores iniciales de temperatura y humedad.
- *Nombre en Arduino*.attachPop (*función, &nombre en Arduino*): este comando define a qué función acudir cuando es soltado un botón.
- pinMode (pin, salida/entrada): con esta instrucción configuramos el pin que queremos como entrada o salida según si queremos recibir información o enviarla.
- digitalWrite (pin, LOW/HIGH): con esta instrucción podemos poner el pin que queramos a nivel alto (5 V) o bajo (0 V).

Ahora se va a proceder a programar el bucle loop de Arduino. Este bucle es repetido continuamente por el microcontrolador, y es, por lo tanto, donde va nuestra programación principal de control y de cálculo del sistema. En la ilustración 38 podemos ver el bucle loop al completo.

```
void loop()
{
  nexLoop(nex_listen_list);
  delay(1000);

  x=funcion_distancia();
  Serial.print("n1.val=");
  Serial.print(x);      //Enviamos serialmente el valor de la distancia
  Funcion_ff();
  delay(750);           //Hacemos una pausa de 750ms

  if ( millis()-auxMillis > intervaloMedidas) {
      h = dht.readHumidity();                          // Leemos Humedad.
      t = dht.readTemperature();                      // Leemos temperatura
      imprimeNextion();
 }
  if(x>=3 && x<=6){
      delay(4000);
      cont=cont+1;
      x=funcion_distancia();
      Serial.print("n2.val=");
      Serial.print(cont);      //Enviamos serialmente el valor de la distancia
      Funcion_ff();
      while(x>=5 && x<=12){
        digitalWrite(IN3,LOW);
        digitalWrite(IN4,LOW);
        analogWrite(ENB,0);
        break;
      }
  }
```

Ilustración 38: Loop de Arduino

Podemos diferenciar que se divide en tres partes. Una de ellas se ejecutará como un proceso normal y las otras dos mediante condiciones if.

- nexLoop (nex_listen_list): consulta los estados de los botones de la pantalla, por si ha sido pulsado alguno de ellos.
- delay (milisegundos): es una función que hace que el procesador espere. Esta espera permite no hacer nada y esperar hasta la ejecución de la siguiente instrucción durante un retardo de tiempo definido.
- Bloque distancia: en este bloque empezamos metiendo la función distancia dentro de la variable global x. Después enviamos serialmente en el display de la pantalla “n1.val=” y seguido la variable x y la función ff que se encarga de enviar los tres valores 0xff.
- Bloque sensor temperatura: empieza con una condición “if” que nos compara que millis (devuelve el número de milisegundos desde que Arduino se ha reseteado) sea mayor que intervaloMedidas. Una vez entre en la condición meteremos en las variables t la temperatura y h la humedad y seguido llamamos a la función imprimeNextion () que mandará los datos por puerto serie a los displays correspondientes de la pantalla.
- El último bloque está diseñado para que en caso de que algún objeto se situé en el sensor HC-SR04 más de 3 segundos (se supone que se ha producido un atranque en la cinta por lo que el objeto quedaría parado ahí) la cinta debe pararse en seco para quitar el atranque de la maquina. También realizaremos el contaje de objetos que han pasado por la cinta. Para ello primero, hacemos uso de una condición if que nos medirá la distancia en la que queremos que este comprendido la distancia del objeto el tiempo que deseemos. Cuando cumpla esa condición metemos la distancia en la variable x para después imprimirla en pantalla y acto seguido aumentamos en uno la variable cont para enviarla serialmente a la pantalla. Por último comprobamos con un bucle “while” que si sigue la distancia comprendida entre el rango que

hemos puesto realice el pare del motor haciendo así que la cinta se detenga ya que el objeto sigue estando obstaculizado.

1.8.9 MAQUETA CINTA TRANSPORTADORA

Para continuar el proyecto se ha diseñado una maqueta de una cinta transportadora para así poder plasmar todos los procesos anteriormente nombrados. A continuación explicare como se ha realizado el montaje.

Primero cogemos dos tubos de PVC de 6cm de largo por 2,5cm de diámetro.

Ilustración 39: Tubos PVC

A continuación cortamos 4 trozos de panel con forma de círculo de 3,5 cm de diámetro.

Ilustración 40: Circunferencias panel

Una vez ambos componentes cortados, procedemos a pegar cada círculo en cada una de los extremos de los tubos quedando como muestra la imagen 41.

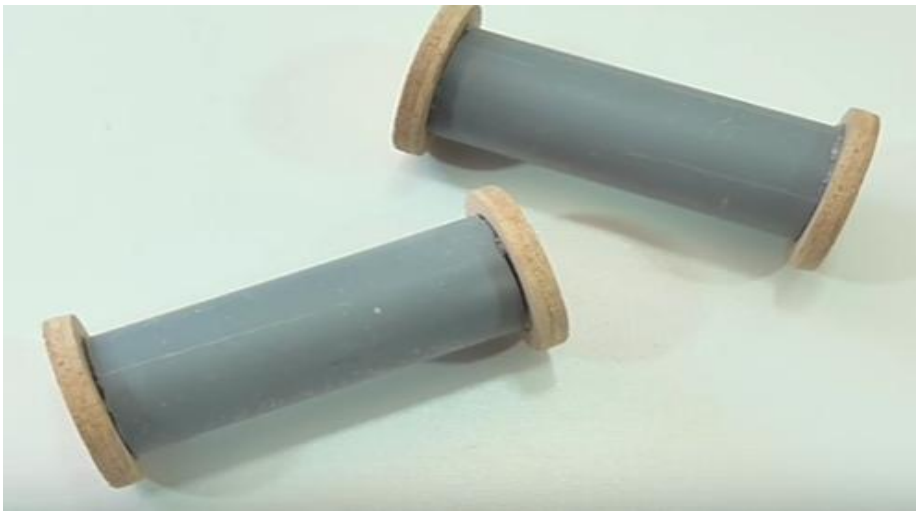


Ilustración 41: tubo con círculos pegados

Ahora introduciremos dos varillas (como las de la imagen 42) en el centro de cada tubo haciéndolas pasar hacia el otro extremo del tubo. Estas tendrán la función de ejes de cada uno de los tubos de PVC. Uno de ellos será eje libre y el otro el que se conectara al motor DC. La Ilustración 43 nos muestra como quedan ambos tubos.

Ilustración 42: Varillas metálicas

Ilustración 43: Tubos PVC con ejes

Ahora cortáremos dos trozos de madera de de 30 cm de largo por 2 cm de ancho. En cada uno de los extremos abriremos un agujero. Cada agujero debe tener la misma distancia (11mm desde el extremo y 2mm desde la parte inferior) desde el extremo hasta el punto donde queremos trabajar. En estos agujeros se introducirá los rodillos con los ejes ya incluidos.

Ilustración 44: Trozos madera

En la siguiente ilustración 45 vemos como quedan los agujeros que hemos hecho en los trozos de madera.

Ilustración 45: Trozos madera con los agujeros en 11mm x 2mm

Ahora cortaremos un trozo de madera de 24 x 7,1 cm. Este se pegara de canto (por el lado de 24 cm) a uno de los trozos de madera con los agujeros.

En la siguiente imagen 46 se puede ver por donde se van a pegar en ambos trozos.

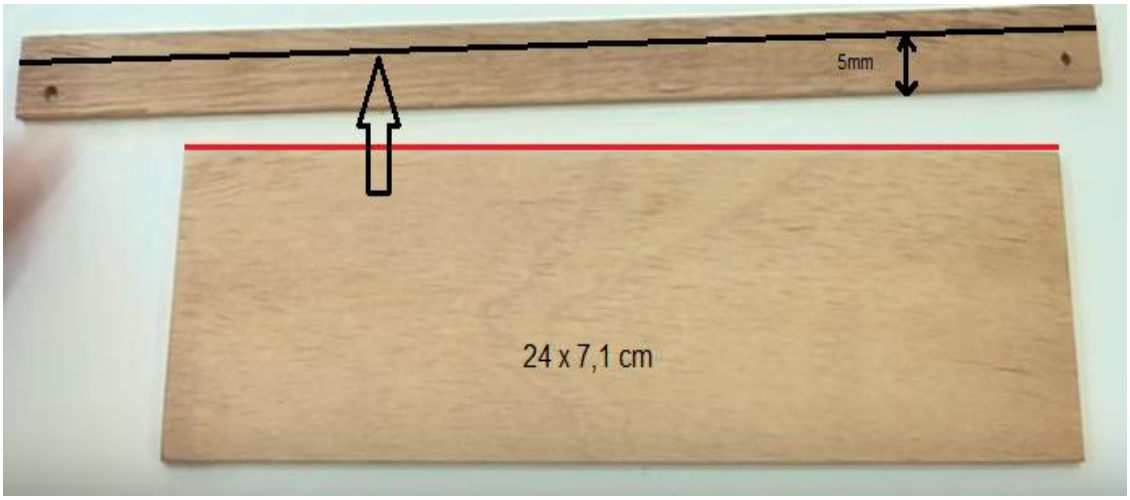


Ilustración 46: Proceso unión ambos trozos

En la siguiente imagen 47 ya se puede ver como quedaría ambos trozos unidos y la introducción de uno de los rodillos. Para ello metemos el eje en uno de los agujeros que hicimos en los trozos.

Ilustración 47: Unión maderas y rodillo

A continuación metemos la otra tira de madera en la parte del eje que queda suelto. También se fijara en el trozo de madera de 24 x 7,1cm.

Ilustración 48: Unión maderas y rodillo (2)

Este mismo proceso se realizara en el otro lado que nos queda libre. En este caso metemos el rodillo que nos queda en los dos agujeros que tenemos en los trozos de madera. Una vez metido procederemos a crear la cinta. Para ello haremos uso de material goma eva. Las dimensiones que debe tener la cinta es de 64 x 6 cm. A continuación, pegaremos con súper glue ambos extremos (como se puede ver en la imagen 49). Todo este proceso se realiza con la goma eva metida dentro de los dos rodillos.

Ilustración 49: Unión cinta

A continuación crearemos la base de la maqueta. Las dimensiones serán de 32 x 20 cm. En ella será donde situaremos 4 soportes donde ira la cinta transporta y los componentes que formaran el proyecto (Arduino, pantalla, etc…). Ya que la cinta ira inclinada dos de los soportes serán más altos. Los más bajos tendrán unas dimensiones de 5 x 2 cm, mientras que los más altos

serán de 7 x 2 cm. En la siguiente imagen 50 se muestra la base con los 4 soportes.

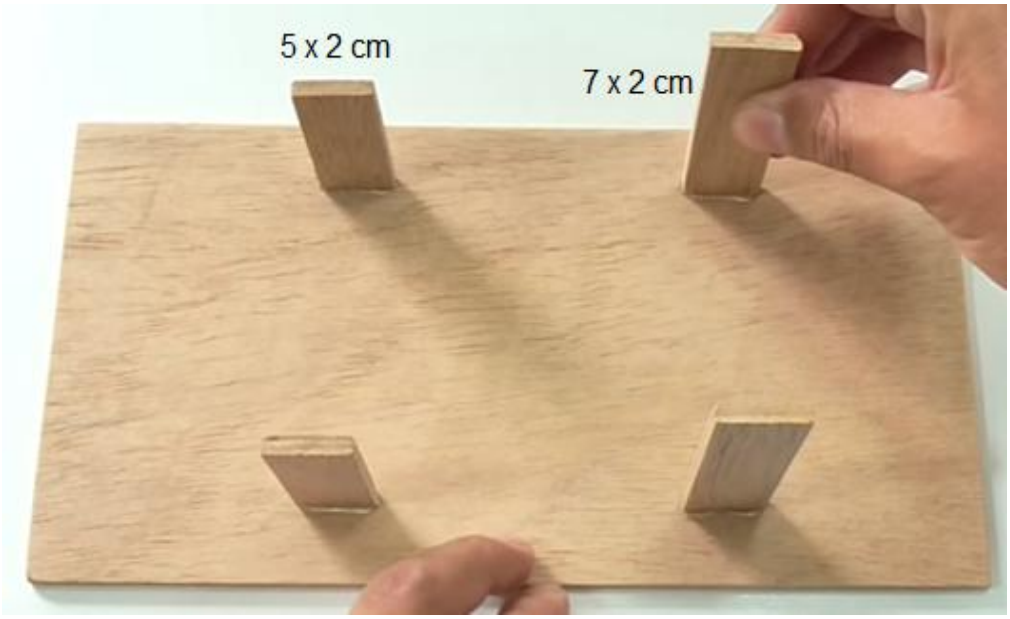


Ilustración 50: Base maqueta

El siguiente paso que realizaremos será meter uno de los engranajes en uno de los dos rodillos (el que queremos que este accionado por el motor). El otro rodillo quedara libre. Ya que el engranaje tiene un diámetro mayor que nuestro eje lo fijaremos con la pistola. En la siguiente imagen 51 puede verse como queda fijado.

Ilustración 51: Engranaje con rodillo

Este mismo paso lo hacemos con el motor dc. Antes de unir el bloque de la cinta a las 4 partes, comprobaremos de qué manera el engranaje del motor y de la cinta tiene buena unión para que el mecanismo funcione correctamente. Una vez tengamos ya la altura optima uniremos el bloque de la cinta y el motor a la base del panel. En la ilustración 52 se puede ver cómo deben de quedar unidos.

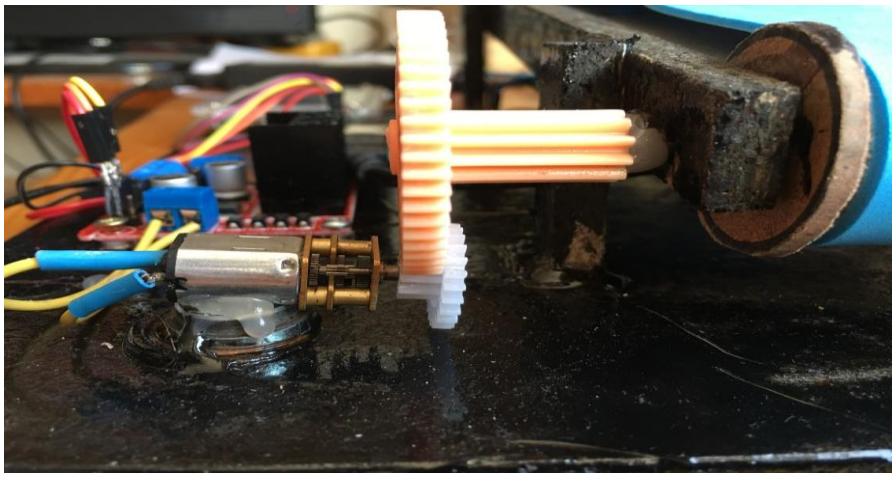


Ilustración 52: Unión motor y cinta

Por último hemos pintado la madera de color negro y colocado cada uno de los componentes que vamos a utilizar en el proyecto. En la siguiente ilustración 53 muestra como ha quedado la maqueta.

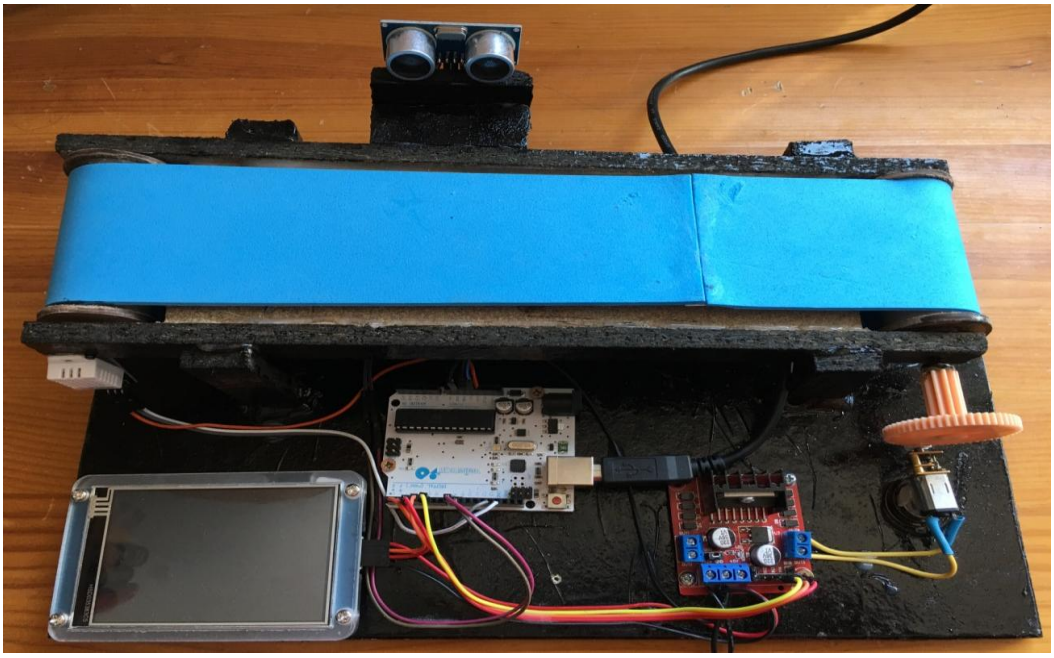


Ilustración 53: Resultado final

2. MEMORIA

2.1. ESPECIFICACIONES JUSTIFICATIVAS

En este apartado explicaremos más detalladamente las especificaciones del diseño de los diferentes circuitos, así como la elección de algunos componentes.

2.1.1. PUENTE EN H (MODULO L298N)

Hemos escogido el driver L298N para controlar los motores ya que internamente posee un puente en H. Esta configuración nos permite controlar el giro de un motor, básicamente los diodos actúan como controladores de la polaridad que se le aplica al motor.

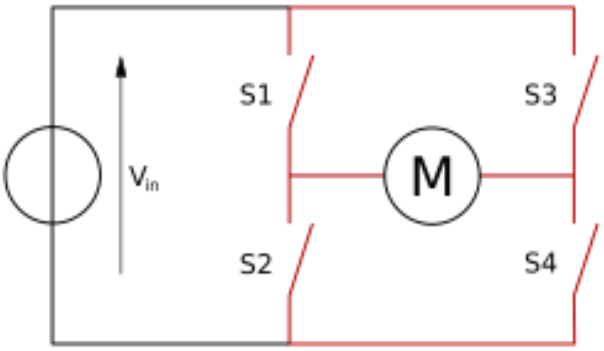


Ilustración 54: Puente en H

Un puente H se construye con 4 interruptores. Cuando los interruptores S1 y S4 (ver figura 55) están cerrados (S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

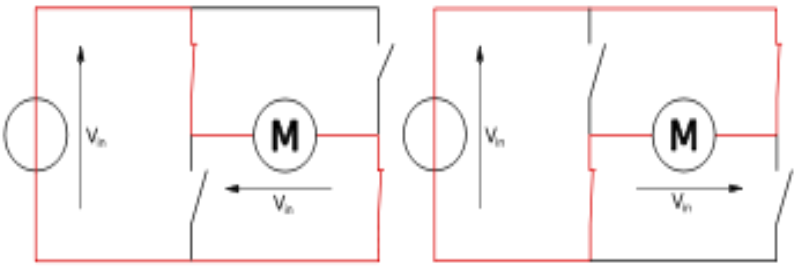


Ilustración 55: Puente en H

De acuerdo a la nomenclatura que estamos usando, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con S3 y S4.

Un puente H también se usa para frenar el giro de un motor (de manera brusca), al hacer un corto entre las bornas del motor, o incluso puede usarse para permitir que el motor frene bajo su propia inercia, cuando desconectamos el motor de la fuente que lo alimenta. En el siguiente cuadro se resumen las diferentes acciones siguiendo como referencia los nombres de los interruptores de la figura 54.

S1	S2	S3	S4	**Resultado**
1	0	0	1	El motor gira en *avance*
0	1	1	0	El motor gira en *retroceso*
0	0	0	0	El motor se detiene bajo su inercia
1	0	1	0	El motor frena (*fast-stop*)
0	1	0	1	El motor frena (*fast-stop*)
1	1	0	0	Corto circuito
0	0	1	1	Corto circuito
1	1	1	1	Corto circuito

Ilustración 56: Control freno motor

2.1.2. MAX232 (SENSOR HC-SR04)

El esquema eléctrico de la placa del sensor de ultrasonidos es el que se muestra en la figura 57. En el podemos observar que está compuesto por tres circuitos integrados. Uno de ellos es el MAX232.

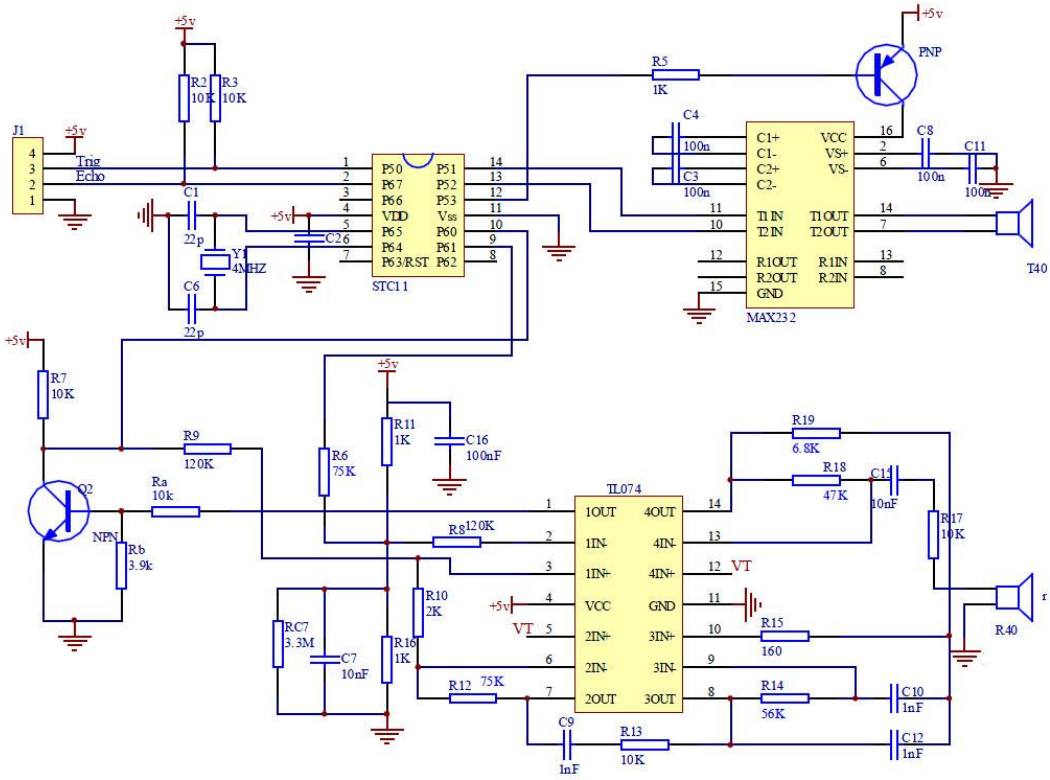
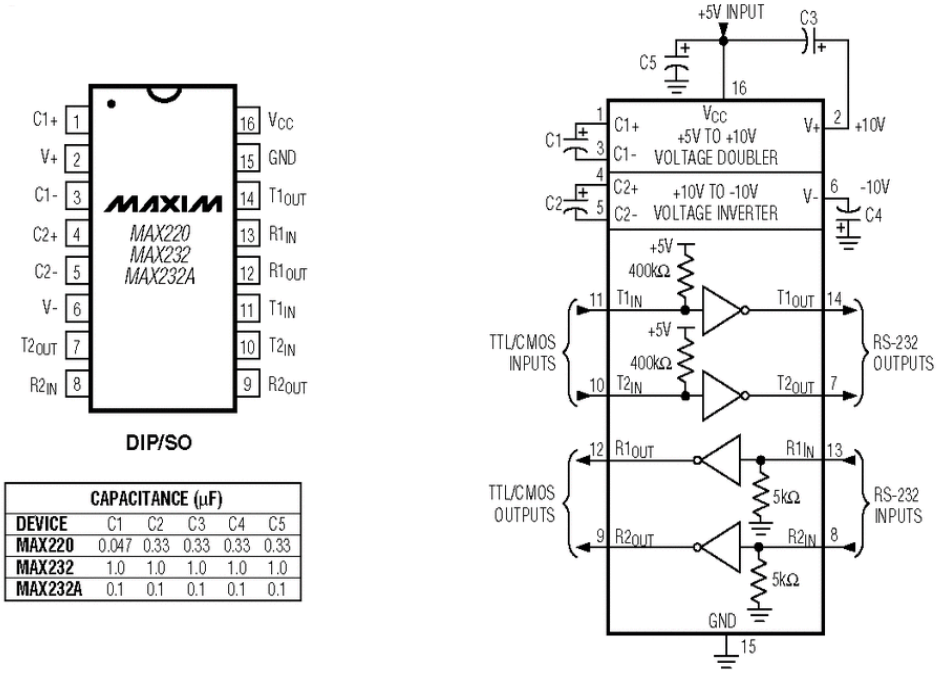


Ilustración 57: Esquema eléctrico sensor HC-SR04

El MAX232 es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. Lo interesante es que sólo necesita una alimentación de 5V, ya que genera internamente algunas tensiones que son necesarias para el estándar RS232.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Ilustración 58: Esquema MAX232

2.1.3. TL074 (SENSOR HC-SR04)

Otro de los circuitos integrados que forman este sensor es el amplificador operacional TL074. Estos, son amplificadores operacionales cuádruples con entradas J-FET de alto voltaje y transistores bipolares en un circuito integrado monolítico. Estos dispositivos cuentan con altas velocidades, polaridad de entrada baja y corriente offset, bajo coeficiente de tensión de compensación de temperatura.

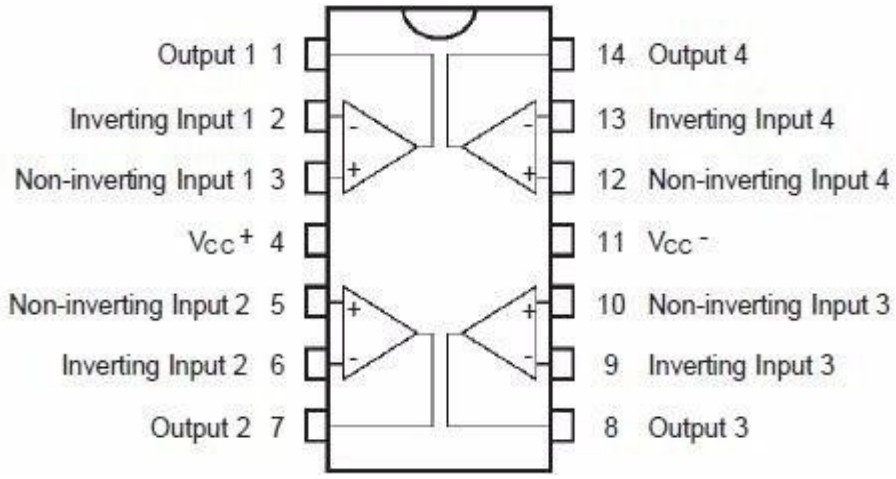


Ilustración 59: Amplificador ST074

El amplificador operacional TL074 tiene un nivel medio de ruido de:

$$e_n = 15nV/\sqrt{HZ}$$

Y se empleará para amplificar una señal ultrasónica de entre 45 y 55KHz, con una salida de 0dBV (1V) y una ganancia de 40dB.

La raíz del ancho de banda es:

$$\sqrt{55e3 - 45e3} = 100$$

El ruido equivalente de entrada será:

$$Ein = 15 * 100 = 1500\ nV$$

Para obtener el ruido de salida se debe multiplicar por la ganancia de 40dB:

$$1500\ nV * 100 = 150\ \mu V$$

La relación señal/ruido (dB):

$$20 * log \frac{1}{150\ \mu V} = 76.48\ dB$$

2.1.4. STC11 (SENSOR HC-SR04)

El corazón del módulo es el microcontrolador STC11F02E de STC, un fabricante de chips chino que produce derivados mejorados del clásico 8051. Es prácticamente donde se procesa toda la información del sensor. El STC11 es un microcontrolador de un solo chip basado en una CPU 80C51 de arquitectura 1T de alto rendimiento, con un kernel capaz de ejecutar instrucciones hasta siete veces más rápido que los dispositivos estándar 8051.

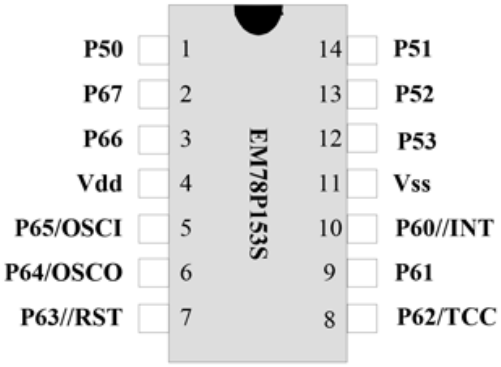


Ilustración 60: STC11

El STC11F02E admite la programación en el sistema (ISP) y la programación en aplicaciones (IAP) y tiene un conjunto de instrucciones que es totalmente compatible con la serie de microcontroladores estándar 80C51.

Este integrado responde al siguiente diagrama de funciones:

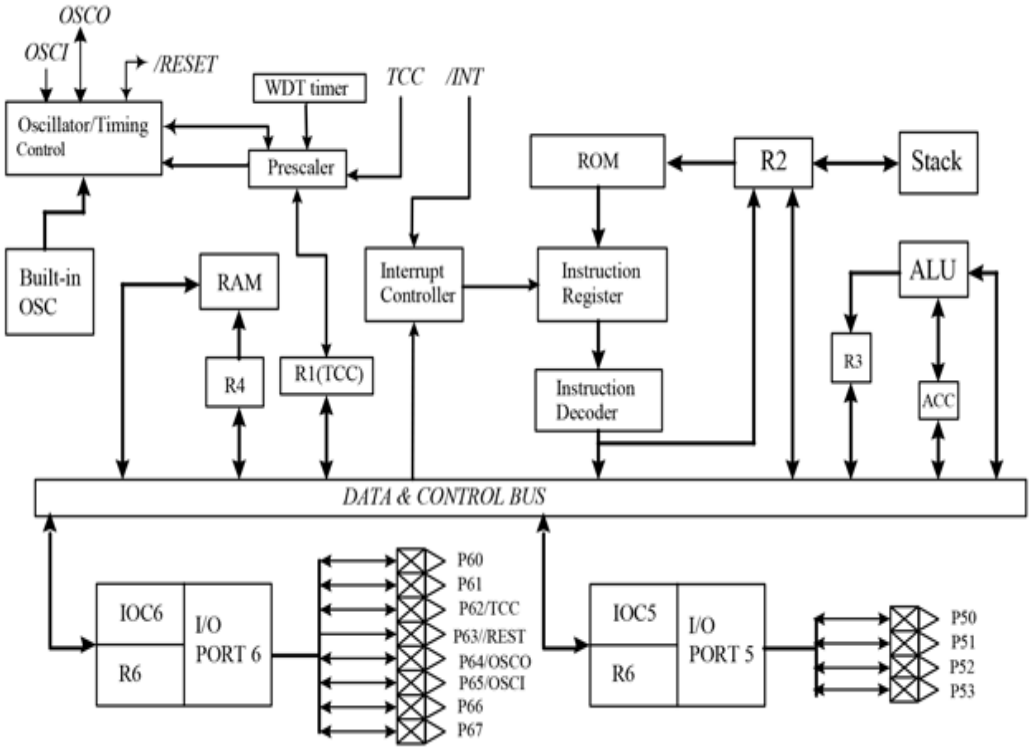


Ilustración 61: Esquema stc11

2.1.5. DHT22 (PROTOCOLO COMUNICACIÓN)

El sensor DHT22 originalmente tiene 4 puertos de los cuales solo usaremos 3 para conectarlo a nuestra Arduino tal y como nos indica el fabricante.

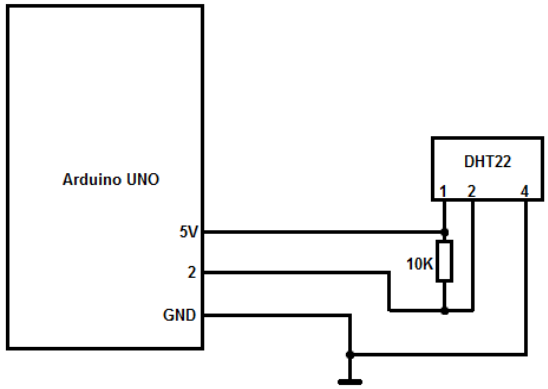


Ilustración 62: Esquema de conexión del DHT22 con un MCU

La MCU se comunicará con el DHT y éste le enviará 5 bytes de datos donde los 2 primeros corresponderán a la Humedad Relativa, los 2 siguientes a la Temperatura y el último byte para el Checksum.

Protocolo de comunicación

Primero de todo, el MCU mandará una secuencia de START al DHT y después el mismo esperará a que el DHT responda. Después el DHT mandará una señal al MCU para indicar que está preparada para inicial la trasmisión bit a bit.

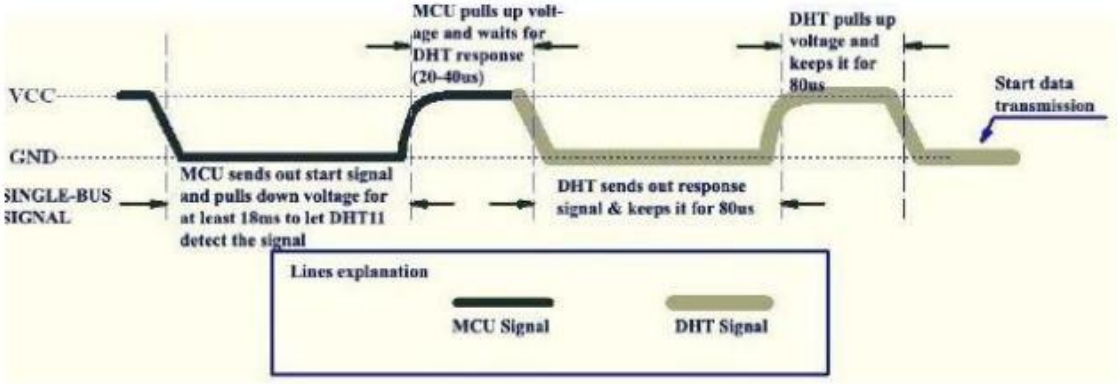


Ilustración 63: Secuencia de START para comunicación con DHT22.

Siempre que inicia la trasmisión de un bit el sensor, mandará un pulso de bajo voltaje de 50us y después enviará un pulso de alto voltaje que según la duración del mismo el pulso significará que es un "0" (26-28us Figura 64) o un "1" (70us Figura 65).

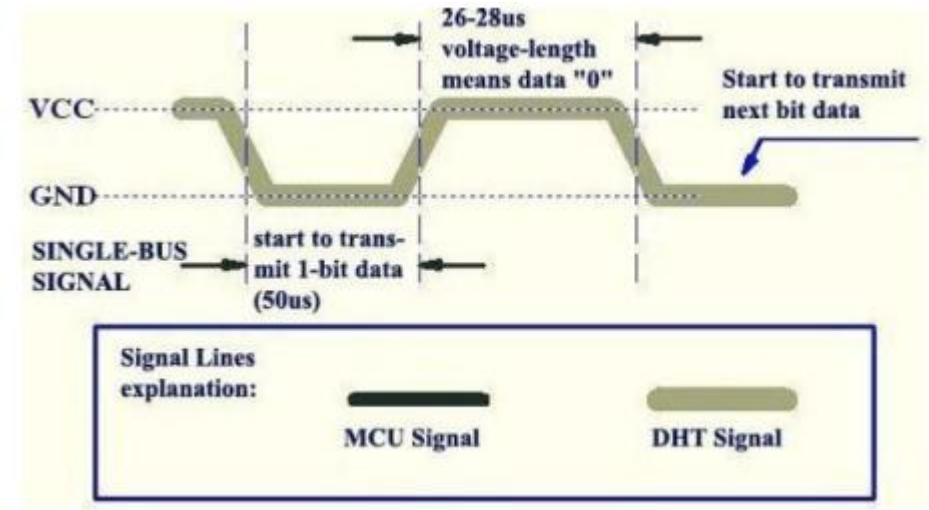


Ilustración 64: Secuencia del DHT22 para enviar un "0".

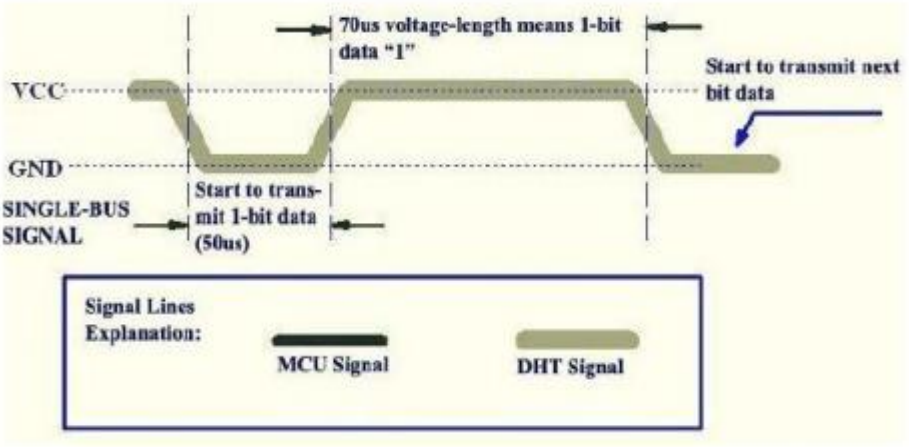


Ilustración 65: Secuencia del DHT22 para enviar un "0".

2.1.6. REGULADOR TENSIÓN NCP1117 (ARDUINO)

Para que nuestra placa Arduino UNO funcione debemos alimentarla al menos con 7V. Por este motivo, nuestra placa Arduino cuenta con un regulador de tensión, cuya función es convertir el voltaje de alimentación al voltaje de los elementos electrónicos, principalmente el microcontrolador. Esta es la parte del circuito que regula una salida a 5V desde el jack de alimentación.

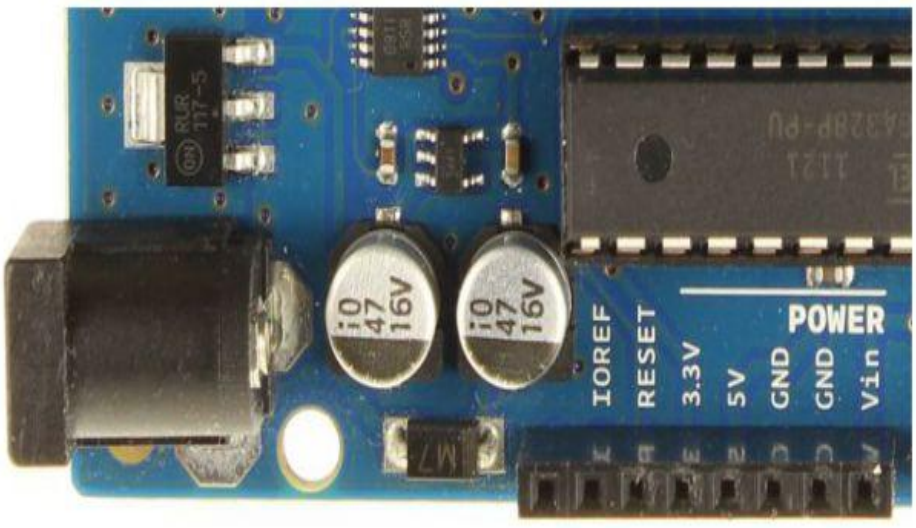


Ilustración 66: Jack alimentación Arduino

Como se puede ver en este esquema (figura 66), Arduino es alimentado mediante un regulador de tensión, en este caso es un NCP1117 que alimenta al bus de 5V de Arduino.

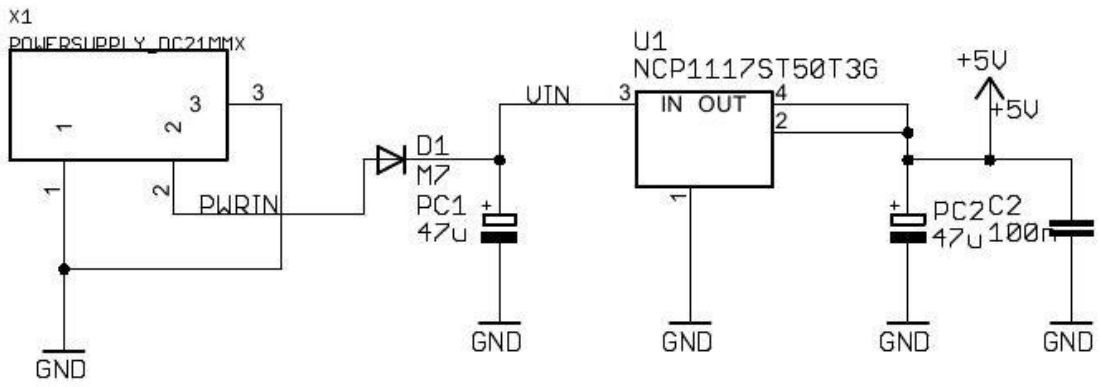


Ilustración 67: Alimentación Arduino

A su vez el bus de 5V alimenta otro regulador de tensión LP2985-33 del que se obtiene una salida de 3.3V.

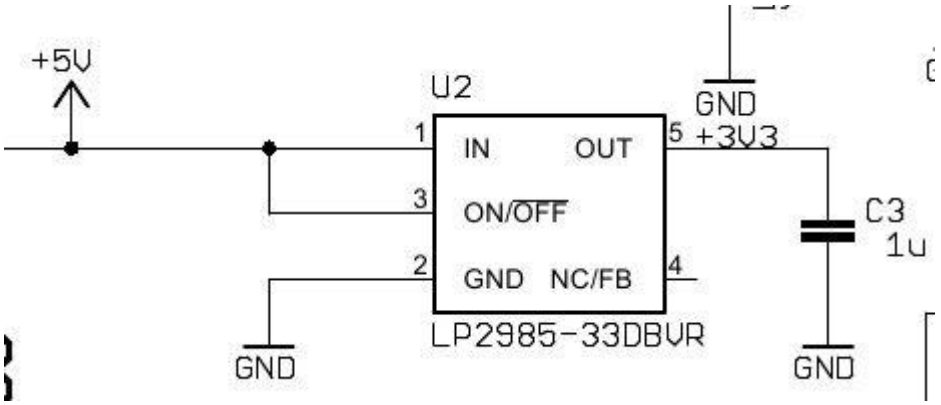


Ilustración 68: Alimentación Arduino

Además hay disponible una entrada al bus de 5V para la alimentación directa del cable USB.

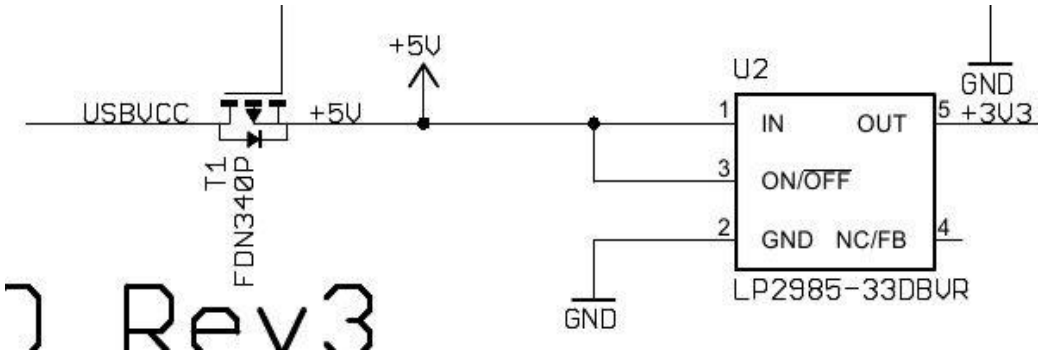


Ilustración 69: Alimentación Arduino

La placa Arduino no funcionará fiablemente con menos de 7V, pero, dado que internamente funciona a 5V, la eficiencia máxima de la alimentación será del 71% (=5V/7V). Si alimentamos la placa a 9V la eficiencia cae hasta el 55%.

2.1.7. MICROCONTROLADOR ATMEGA328 (ARDUINO)

En este apartado explicaremos más detalladamente el funcionamiento del microcontrolador de la placa Arduino UNO el ATMEGA328. Toda la información del micro la podemos encontrar en su datasheet que consta de 444 páginas. Nosotros explicaremos las características más importantes.

Los parámetros del microcontrolador son:

PARÁMETROS	VALORES
Flash	32 Kbytes
SRAM	2 Kbytes
Cantidad Pines	28
Frecuencia máxima de operación	20 MHz
CPU	8-bit AVIR
Pines máximos de E/S	23
Interrupciones internas	24
SPI	1
UART	1
Canales ADC	8
Resolución de ADC	10
Eeprom	1K
Canales PWM	6
Voltaje de operación	1.8-5.5 v
Timers	3

Ilustración 70: Parámetros microcontrolador

El diagrama de bloques fundamental de un microcontrolador se compone básicamente de tres bloques: CPU, memoria (RAM y ROM) y las entradas/salidas. Los bloques se conectan entre sí mediante grupos de líneas eléctricas denominadas buses o pistas. Los buses pueden ser de direcciones o de datos. La CPU también incluye los circuitos para realizar operaciones aritméticas y lógicas elementales con los datos binarios, en la denominada Unidad Aritmética y Lógica (ALU). En la siguiente imagen 71 se puede ver el diagrama de bloques de nuestro microcontrolador

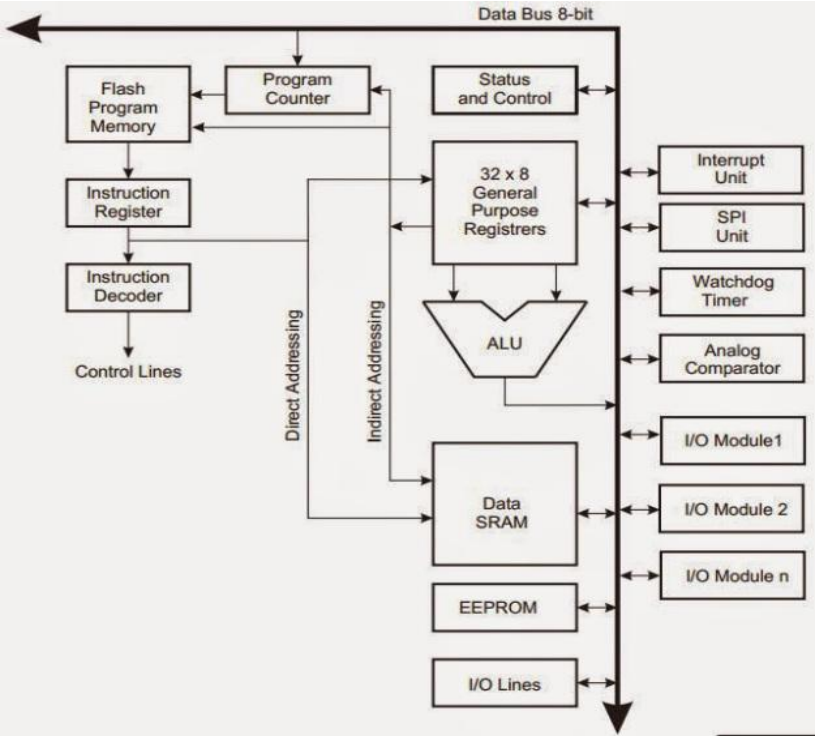


Ilustración 71: Diagrama bloques microcontrolador

Cuando el microcontrolador ejecuta una instrucción, internamente hace muchas operaciones y cada una de esas operaciones se ejecuta en un ciclo de reloj. Para el ATmega328 que tiene una frecuencia de 16 MHz, es decir, cada ciclo tarda 0,0000000625 segundos = 0,0625 microsegundos = 62,5 nanosegundos.

El microcontrolador ATmega328 tiene tres timers (timer 0, timer 1, timer 2) que también se pueden usar como contadores. Los timers 0 y 2 son de 8 bits y el timer 1 de 16. Son módulos que funcionan en paralelo a la CPU y de forma

independiente a ella. El funcionamiento básico consiste en aumentar el valor del registro del contador al ritmo que marca su señal de reloj.

La memoria RAM en el ATmega328 se divide en varias partes, todos los grupos de registros se ponen a cero al apagar la fuente de alimentación. La SRAM del 328 se distribuye de la siguiente forma:

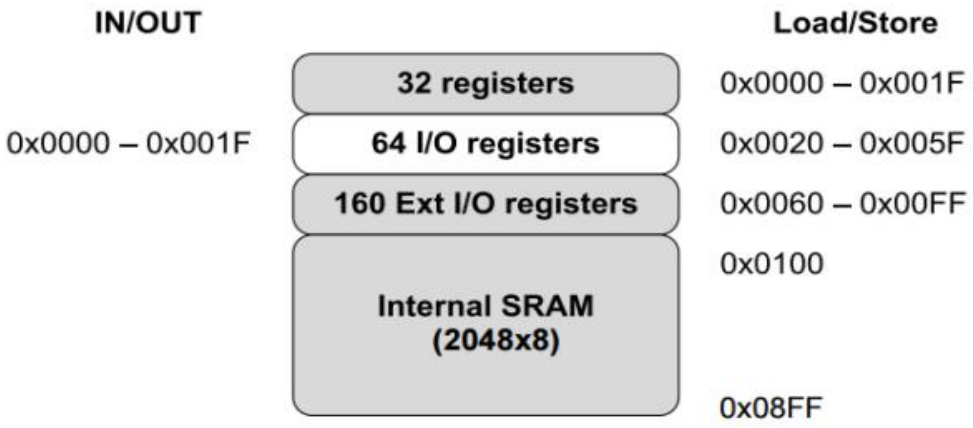


Ilustración 72: Memoria RAM ATMEGA328

2.1.8. MICROCONTROLADOR ATMEGA16U2 (ARDUINO)

Si nos fijamos en el pequeño integrado que hay en la placa de Arduino UNO junto al conector USB, se trata de un ATmega16u2 cuya misión es dar el interfaz USB al Arduino UNO y comunicar los datos con el ATmega328 mediante el puerto serie. Se podría usar como microcontrolador completamente funcional y no solo un conversor de USB a Serial con ciertas modificaciones. Podríamos usar ambas MCUs en la misma placa, pudiendo descargar trabajo de la MCU principal en la secundaria.

Para ello usa el hoodloader2 en el Atmega16U2 dependiendo de la versión de Arduino Uno que tengamos y comunicamos ambas MCUs por HW serial

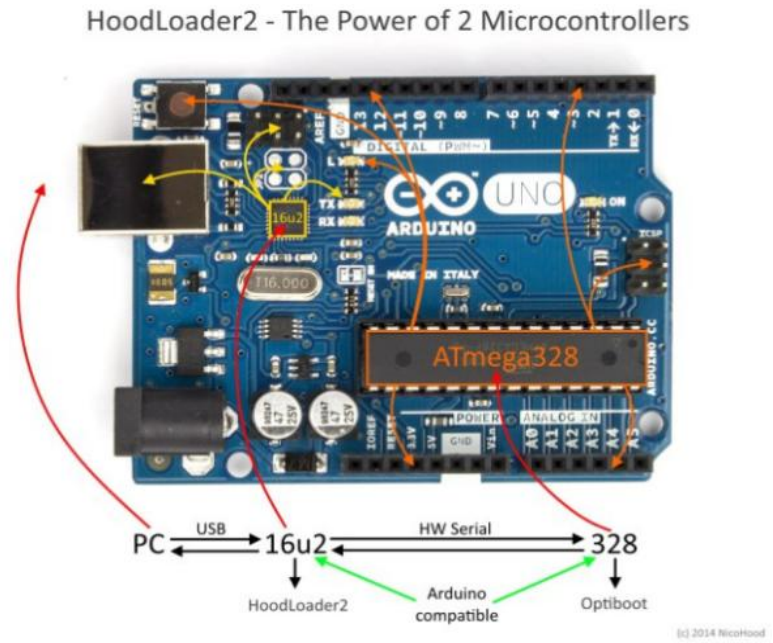


Ilustración 73: Distribución micro ATMEGA16U2 Arduino

2.1.9. CONVERSOR ANALOGICO/DIGITAL

Un conversor analógico-digital es un dispositivo electrónico capaz de convertir una señal analógica en un valor binario, en otras palabras, éste se encarga de transformar señales analógicas a digitales (0 y 1).

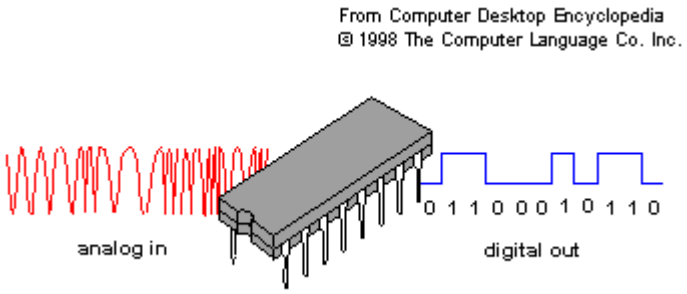


Ilustración 74: Conversor A/D

La resolución determina la precisión con la que se reproduce la señal original. Esta resolución se pude saber, siempre y cuando conozcamos el valor máximo de la entrada a convertir y la cantidad máxima de la salida en dígitos binarios.

$$Resolucion = +\frac{Vref}{2^n} (donde\ n\ es\ el\ n^{o}bits)$$

La tarjeta Arduino utiliza un conversor A/D de 10-bits, así que: Resolución = Vref/1024 Mapeará los valores de voltaje de entrada, entre 0 y Vref voltios, a valores enteros comprendidos entre 0 y 1023 (2^n-1). Con otras palabras, esto quiere decir que nuestros sensores analógicos están caracterizados con un valor comprendido entre 0 y 1023.

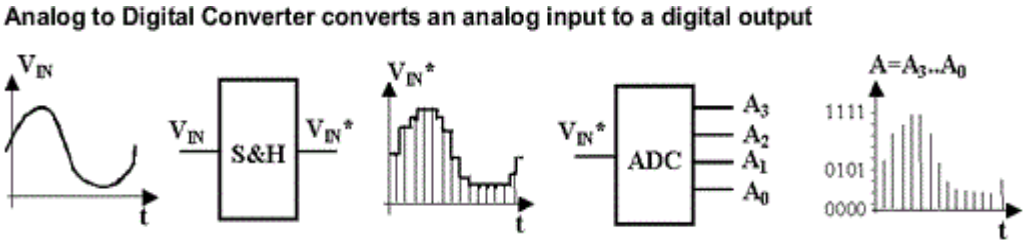


Ilustración 75: Proceso conversión analógico/digital

2.2. CODIGO FUENTE ARDUINO

```
#include <Nextion.h>
#include "DHT.h"
#define DHTPIN 13                                   // Pin al que se conecta
                                                      la salida del arduino.
#define DHTTYPE DHT22                               // Tipo de sensor que
                                                      vamos usar (DHT 22).

////CONSTANTES GLOBALES PARA EL CONTROL MOTOR///////////////
int IN3 = 5;    // Input3 conectada al pin 5 de Arduino
int IN4 = 4;    // Input4 conectada al pin 4 de Arduino
int ENB = 3;    // ENB conectada al pin 3 de Arduino
int flag_vel = 0; // flag de velocidad del motor
int vel_pwm = 0;  // flag para el control de pwm
//////////////////////////////////////////////////////////////

////CONSTANTES GLOBALES SENSOR DISTANCIA//////////////////////////
const int Trigger = 9 ;   //Pin digital 7 para el Trigger del sensor
const int Echo = 8;   //Pin digital 8 para el Echo del sensor
unsigned long x=0; // variable global para la distancia
unsigned long cont=0;// variable para el contador de objetos
//////////////////////////////////////////////////////////////

////CONSTANTES GLOBALES TEMPERATURA////////////////////////////////
float h=0; //variable para la humedad
float t=0; // variable para la temperatura
int intervaloMedidas = 1000;// variables para la actualizacion
int auxMillis=0;            // de la temperatura y humedad
//////////////////////////////////////////////////////////////
//definición de los botones (pagina, id, nombre)//
NexButton b0 = NexButton(2, 2, "b0");
```

```
NexButton b1 = NexButton(2, 3, "b1");
NexButton b3 = NexButton(2, 5, "b3");
NexButton b4 = NexButton(2, 6, "b4");
////////////////////////////////////////////////

//definicion de los numeros (pagina, id, nombre)//
NexNumber n0 = NexNumber(2,1,"n0");
NexNumber n1 = NexNumber(3,2,"n1");
NexNumber num0 = NexNumber(4,7,"num0");
NexNumber num1 = NexNumber(4,8,"num1");
////////////////////////////////////////////////

//Funcion para consultar el estado de los botones//
NexTouch *nex_listen_list[] =
{
  &b0,
  &b1,
  &b3,
  &b4,
  &n0,
  &n1,
  &num0,
  &num1,
  NULL
};
////////////////////////////////////////////////////

DHT dht(DHTPIN, DHTTYPE);                          // Inicializa el sensor.

void setup()
{
 Serial.begin(9600);
 nexInit();
 dht.begin();                                       // Iniciamos sensor.
 imprimeNextion();
 b0.attachPop(pulsador_mas);
 b1.attachPop(pulsador_menos);
 b3.attachPop(pulsador_start);
 b4.attachPop(pulsador_stop);
 pinMode (ENB, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (IN4, OUTPUT);
 pinMode(Trigger, OUTPUT);                          //pin como salida
 pinMode(Echo, INPUT);                             //pin como entrada
 digitalWrite(Trigger, LOW);                       //Inicializamos el pin con 0
}


void pulsador_mas(void *ptr)          //función para elevar la velocidad del motor//
{
```

```
  if(vel_pwm > 255){
    vel_pwm=255;
  }else{
      flag_vel=flag_vel+5;
      vel_pwm = flag_vel;
  }
  Serial.print("n0.val=");
  Serial.print(vel_pwm);
  Funcion_ff();
  if(vel_pwm <= 255 && vel_pwm >= 0){
      digitalWrite(IN3,HIGH);
      digitalWrite(IN4,LOW);
      analogWrite(ENB,vel_pwm);
  }
}

void pulsador_menos(void *ptr)              //funcion para reducir la velocidad//
{
  if(flag_vel <= 0){
    vel_pwm=0;
  }else{
      flag_vel=flag_vel-5;
      vel_pwm = flag_vel;
  }
  Serial.print("n0.val=");
  Serial.print(vel_pwm);
  Funcion_ff();
  if(vel_pwm >= 0 && vel_pwm <=255){
      digitalWrite(IN3,HIGH);
      digitalWrite(IN4,LOW);
      analogWrite(ENB,vel_pwm);
  }
}

void pulsador_stop(void *ptr)               //Para inmediatamente el motor//
{
  flag_vel=0;
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,0);
  Serial.print("n0.val=");
  Serial.print(flag_vel);
  Funcion_ff();
}



void pulsador_start(void *ptr)      //Arranca el motor a una velocidad de 60 rpm//
{
```

```
  flag_vel=80;
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,flag_vel);
  Serial.print("n0.val=");
  Serial.print(flag_vel);
  Funcion_ff();
}

void Funcion_ff(void)                    // Función utilizada para la comunicación
{
Serial.write(0xff);
Serial.write(0xff);
Serial.write(0xff);
}


//funcion medida de la distancia//
unsigned long funcion_distancia(){

  long t; //timepo que tarda en llegar el eco
  long d; //distancia en centimetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);                          //Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH);              //obtenemos el ancho del pulso
  d = t/59;                             //escalamos el tiempo a una distancia en cm
  return d;
}

void imprimeNextion () {
  int auxhumedad = int(h);
  int temperatura = int(t);
  Serial.print("num0.val=");
  Serial.print(temperatura);
  Funcion_ff();
  String humedad = String (auxhumedad);
  Serial.print("num1.val=");
  Serial.print(humedad);
  Funcion_ff();
}

void loop()
{
  nexLoop(nex_listen_list);
  delay(1000);

  x=funcion_distancia();
```

```
  Serial.print("n1.val=");
  Serial.print(x);                    //Enviamos serialmente el valor de la distancia
  Funcion_ff();
  delay(750);                         //Hacemos una pausa de 100ms*/

  if ( millis()-auxMillis > intervaloMedidas) {
       h = dht.readHumidity();                          // Leemos Humedad.
       t = dht.readTemperature();                       // Leemos temperatura
       imprimeNextion();
 }
  if(x>=5 && x<=12){
     delay(4000);
     cont=cont+1;
     x=funcion_distancia();
     Serial.print("n2.val=");
     Serial.print(cont);              //Enviamos serialmente el valor de la distancia
     Funcion_ff();
     while(x>=5 && x<=12){
       digitalWrite(IN3,LOW);
       digitalWrite(IN4,LOW);
       analogWrite(ENB,0);
       break;
     }
  }
}
```

3. ANEXOS

3.1. SIMULACION SENSOR HC-SR04

En este apartado vamos a realizar la simulación del sensor ultrasonidos y que nos muestre la distancia en un display LCD. Para ello haremos uso del programa Proteus y compilaremos el código de programa mediante Arduino.

El primer paso es descargar e instalar la librería en Proteus. Esta incluye tanto el sensor como distintos modelos de tarjetas de Arduino.

Ilustración 76: Librería Proteus

A continuación una vez introducida el modelo de tarjeta de Arduino que queremos, deberemos cargar el fichero .HEX que nos proporciona Arduino con el código que queremos que contenga nuestro programa. Para ello solo deberemos hacer doble click sobre el dispositivo y en la ventana emergente se selecciona la opción UltraSonicSensor.HEX y se selecciona la ruta en donde se encuentre alojado el archivo .HEX.

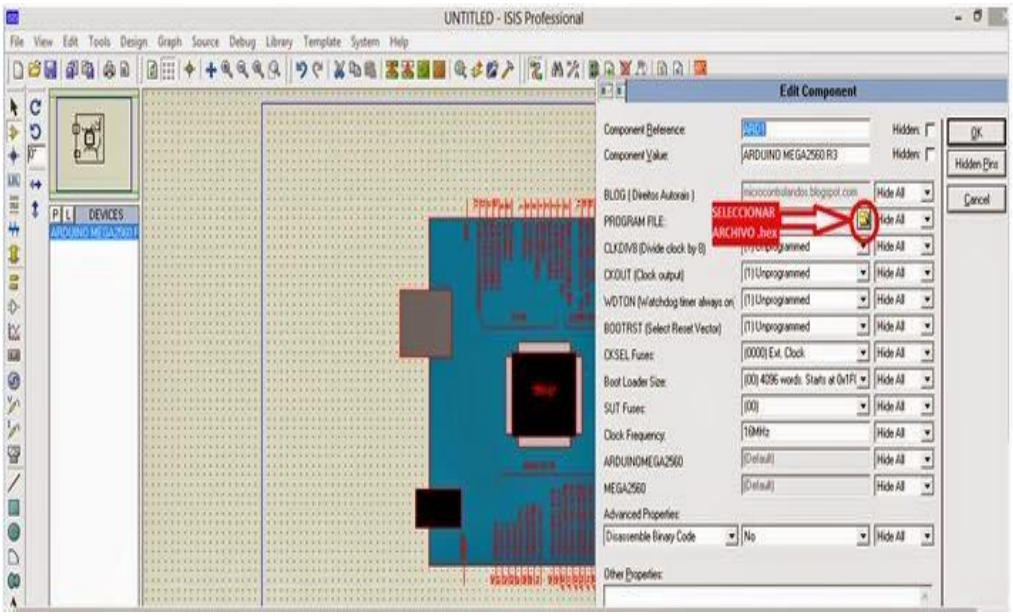


Ilustración 77: Cargar fichero .HEX

Ahora introducimos el modelo del sensor en el área de trabajo, pero también debemos de seleccionar un potenciómetro que hará como divisor de voltaje y con el cual se va simular la variación de distancia. Este debe ir conectado al terminal del sensor llamado SimPin.

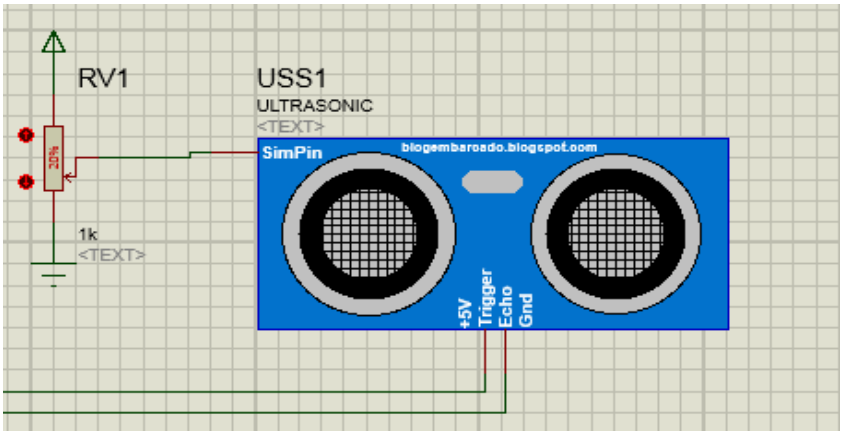


Ilustración 78: Ultrasonidos Proteus

Este modelo de sensor está construido a partir de un microcontrolador que contiene internamente y es el que recibe la variación de voltaje del potenciómetro con lo que se indica una variación de distancia, los otros dos terminales que tiene este sensor deben ser conectados a la tarjeta Arduino. En este caso el Echo al pin 8 de Arduino y el Trigger al pin numero 7.

Para finalizar, a continuación se puede ver en la imagen 79 el proyecto simulado y funcionando correctamente.

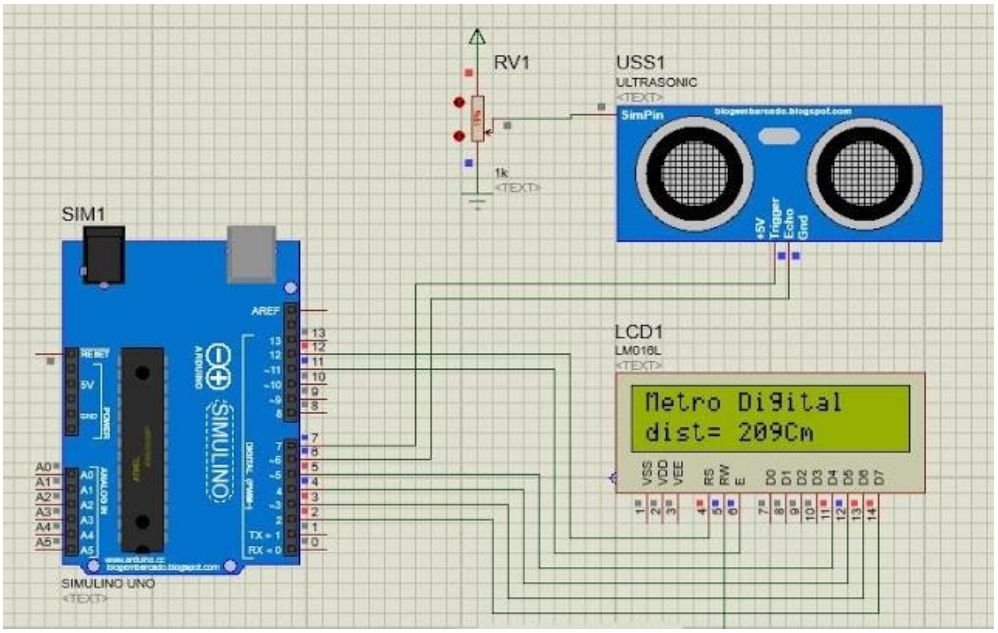


Ilustración 79: Simulación Proteus

4. PRESUPUESTO

4.1. PRECIOS SIMPLES

- Trasformador Arduino 20VA 9.69€
- Trasformador modulo L298n 7.15€
- Tarjeta Arduino UNO 15.96€
- Pantalla Nextion 23.66€
- Engranaje grande 0.55€
- Engranaje pequeño 0.35 €
- Motor dc 12V 7.96€
- Sensor HC-SR04 2.16€
- Sensor DHT22 2.86€

- Modulo L298n 2.95€
- Cable macho 0.15€
- Cable hembra 0.15€
- Goma Eva 0.50€
- Madera maqueta 3.00€
- Tornillos y tuercas 0.10€

4.2. MANO DE OBRA

- Ingeniero Electrónico 12.00€/h
- Corte maqueta 2.00€
- Fijación componentes 3.00€
- Montaje cinta 5.00€

4.3. UNIDADES DE OBRA

- Ud. Distribución Maqueta

Unidad para la realización de la maqueta según las medidas. Una vez tenemos las piezas, el operario debe fijar los componentes en la tabla, mediante tornillos y tuercas, y realizar el proceso de montaje de la cinta.

Descomposición en la siguiente tabla:

TÍTULO	CANTIDAD	PRECIO	TOTAL
Montaje cinta transportadora	1	5.00€	5.00€
Fijación componentes	1	3.00€	3.00€
Tornillo y tuerca	6	0.10€	0.60€
Operario	2,5h	12.00€/h	30.00€
TOTAL			38.60€

Tabla 4.1

- Ud. Conexión electrónica y componentes

Unidad montaje, en la que hay que realizar la conexión da cada componente, según los planos. Deben de cortarse los extremos de los cables para la conexión. Los extremos de ambos transformadores eléctricos deben ser

cortados y adaptados. Una vez realizadas las conexiones, se fijará los cables a la base de la maqueta teniendo una buena distribución. Inserción de los engranajes tanto en el rodillo como en el motor.

TÍTULO	CANTIDAD	PRECIO	TOTAL
Trasformador Arduino 20VA	1	9.69€	9.69€
Trasformador modulo l298n	1	7.15€	7.15€
Tarjeta Arduino UNO	1	15.96€	15.96€
Pantalla Nextion	1	23.66€	23.66€
Engranaje grande	1	0.55€	0.55€
Engranaje pequeño	1	0.35 €	0.35 €
Motor DC 12V	1	7.96	7.96€
Sensor HC-SR04	1	2.16€	2.16€
Sensor DHT22	1	2.86€	2.86€
modulo L298n	1	2.95€	2.95€
Cable macho	10	0.15€	1.50€
Cable hembra	7	0.15€	1.05€
Goma Eva	1	0.50€	0.50€
Madera maqueta	2	3.00€	6.00€
Operario	2.5h	12.00€/h	30.00€
TOTAL			112.34€

Tabla 4.2

4.4. PRESUPUESTO TOTAL

UNIDAD DE OBRA	CANTIDAD	PRECIO	TOTAL
Ud. Distribución maqueta	1	38.60€	38.60€
Ud. Conexión eléctrica y componentes	1	112.34€	112.34€
TOTAL			150.94€

Tabla 4.3

TOTAL: CIENTO CINCUENTA EUROS CON NOVENTA Y CUATRO CÉNTIMOS