Informe Final del Proyecto:

Circuitos Y Mediciones

Carrera: Ingeniería en Informática

Integrantes:

* Calonge Federico.

* Cabot Lucas.

Índice

• 1- Introducción y objetivo del proyecto

Nuestro proyecto consiste en medir la temperatura ambiente mediante un **sensor de temperatura** y de acuerdo a esta, que haya que encender alguno de los **3 leds** (verde, amarillo o rojo) según corresponda y además controlar la velocidad de un **ventilador.**

De esta forma:

-Si se prende el led verde indica una temperatura baja (menor a 18°C), y el ventilador anda a una velocidad baja, funciona a 8,8V (en los cálculos, pero realmente llega a 8,6V) con un ciclo de trabajo (DC) de 35%; de esta forma un 35% del tiempo la señal está activa (el ventilador gira), y un 65% del tiempo inactiva (el ventilador no gira, pero NO se detiene, ya que hubo un 35% del tiempo que estuvo girando).

-Si se prende el led amarillo indica una temperatura "normal" (entre 18°C y 22°C), y el ventilador anda a velocidad media, funciona 8,6V con un DC=50%.

-Si se prende el led rojo indica una temperatura alta (mayor a 22°C), y el ventilador anda a máxima velocidad, funciona a 8,6V con un DC=100% (de esta manera todo el tiempo la señal está activa).

También contará con un **display LCD** que configuramos para mostrar la temperatura. Todo este circuito (que ya veremos en el desarrollo del informe con exactitud cómo está formado y cómo funciona) está alimentado por una fuente de 9V DC que conectamos DIRECTAMENTE a 220V CA. Esta fuente la usamos para alimentar al Arduino y al ventilador. A su vez, el Arduino (que posee internamente un regulador de voltaje de 5V DC) alimenta a distintos componentes: LCD, 3 leds, amplificador no inversor, transistor, y al sensor de temperatura con dichos 5V.

Todo este circuito es "controlado" por software, mediante el entorno de desarrollo (IDE) de Arduino (a excepción de la fuente de 9V DC que es sólo electrónica) que indica qué led se debe prender, por cuanto tiempo, cómo y cada cuanto se toman los datos del sensor, el ancho de pulso que le mando a cada señal (PWM), etc.

▪ 1.1- Cómo nos organizamos y llevamos adelante el proyecto

Para cada parte del circuito total, primero pensamos en los materiales y componentes electrónicos que íbamos a utilizar, viendo los datasheet correspondientes y analizando qué modelo y por qué era el más conveniente. Esto lo fuimos ajustando y mejorando de acuerdo a los problemas que fuimos teniendo y a las mediciones que realizamos. Luego, nos fuimos dividiendo las tareas por clases, con el fin de que en cada clase adelantáramos algo, algún tema específico.

Por lo general, cada clase trabajamos un tema, por ejemplo una clase probamos el transformador y medimos la tensión de salida, otra clase probamos cómo encender un led con un delay (retraso) dado. Luego, lo fuimos complejizando en función de nuestro proyecto. Hubo temas que no

alcanzaban con sólo una clase, así que nos abarcó más clases y tiempo fuera de horario de cursada.

▪ 1.2- Materiales necesarios

A continuación detallaremos todos los materiales necesarios para realizar este proyecto:

- ***Componentes conectados al Protoboard:***

1-Arduino UNO (con microcontrolador ATmega328P).

2-Ventilador (o también llamado Motor trifásico ó Cooler) KD1209PTS1 H.

3-Cables macho a macho para pruebas electrónicas, de distintos colores (generalmente rojos para indicar tensiones y negros para indicar tierra/GND). Y un conector power hembra 220V CA.

4-Display LCD 16x2 GMD 1602k.

5-Sensor de temperatura LM35.

6-3 LEDS (verde, amarillo, rojo).

7-Transistor NPN BD437.

8-Amplificador Operacional No inversor CA3140EZ

9-Resistencias: 1 de 4.7kohm, 3 de 320ohm, 1 de 560ohm y 1 de 470ohm.

10-Potenciomentro 1k

- ***No conectados al Protoboard:***

11-Fuente lineal 9V; que consiste en los siguientes componentes (los cuales detallaremos el conexionado de estos en las páginas 6-11):

-Fusible.

-Llave a un punto.

-Inductor.

-Transformador de entrada 220V 60Hz y salida 9V 1A.

-4 diodos LN4007.

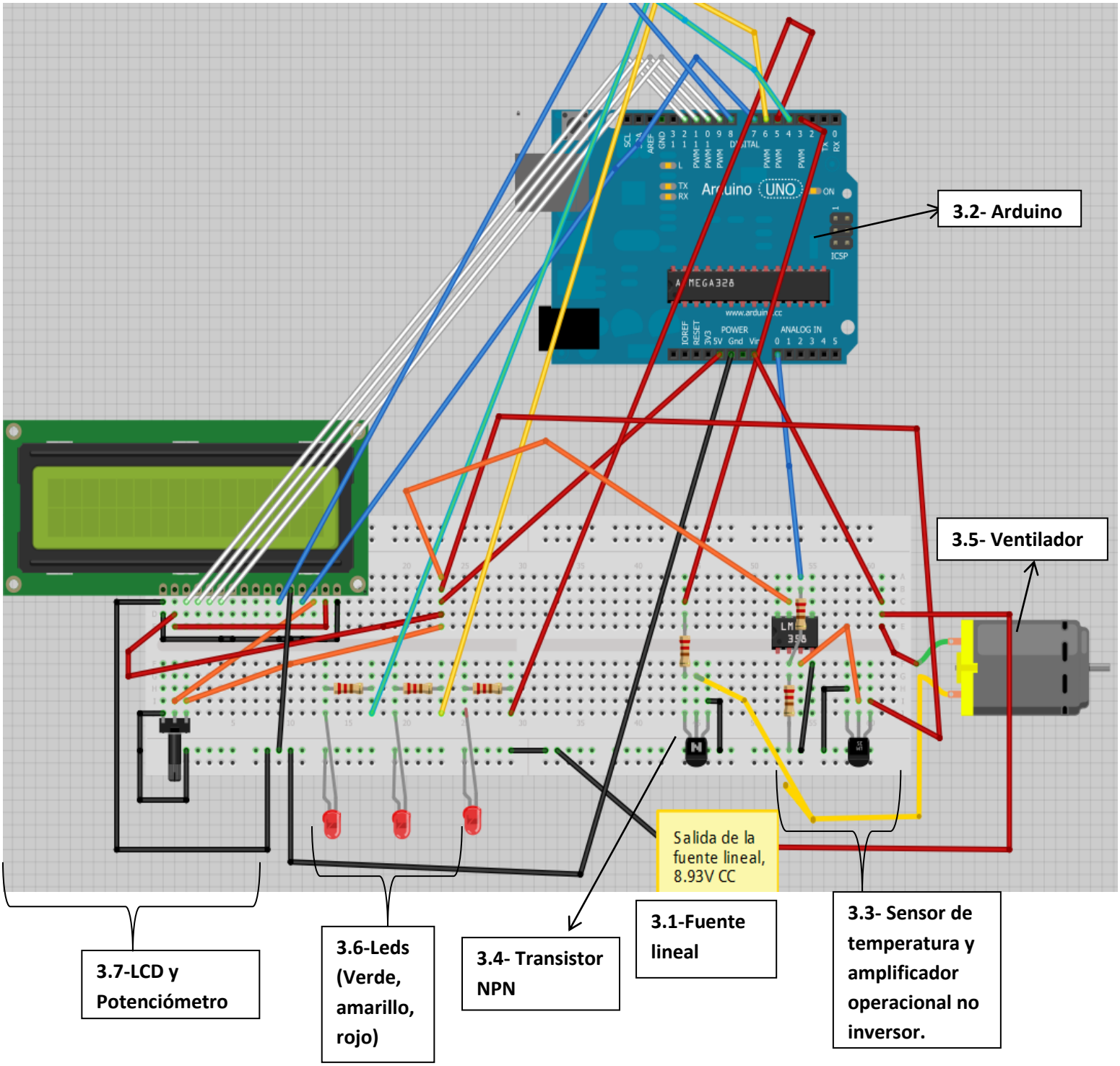
-Capacitor electrolítico de 1000uF 35V.

-Capacitor de cerámica de 0.1uF.

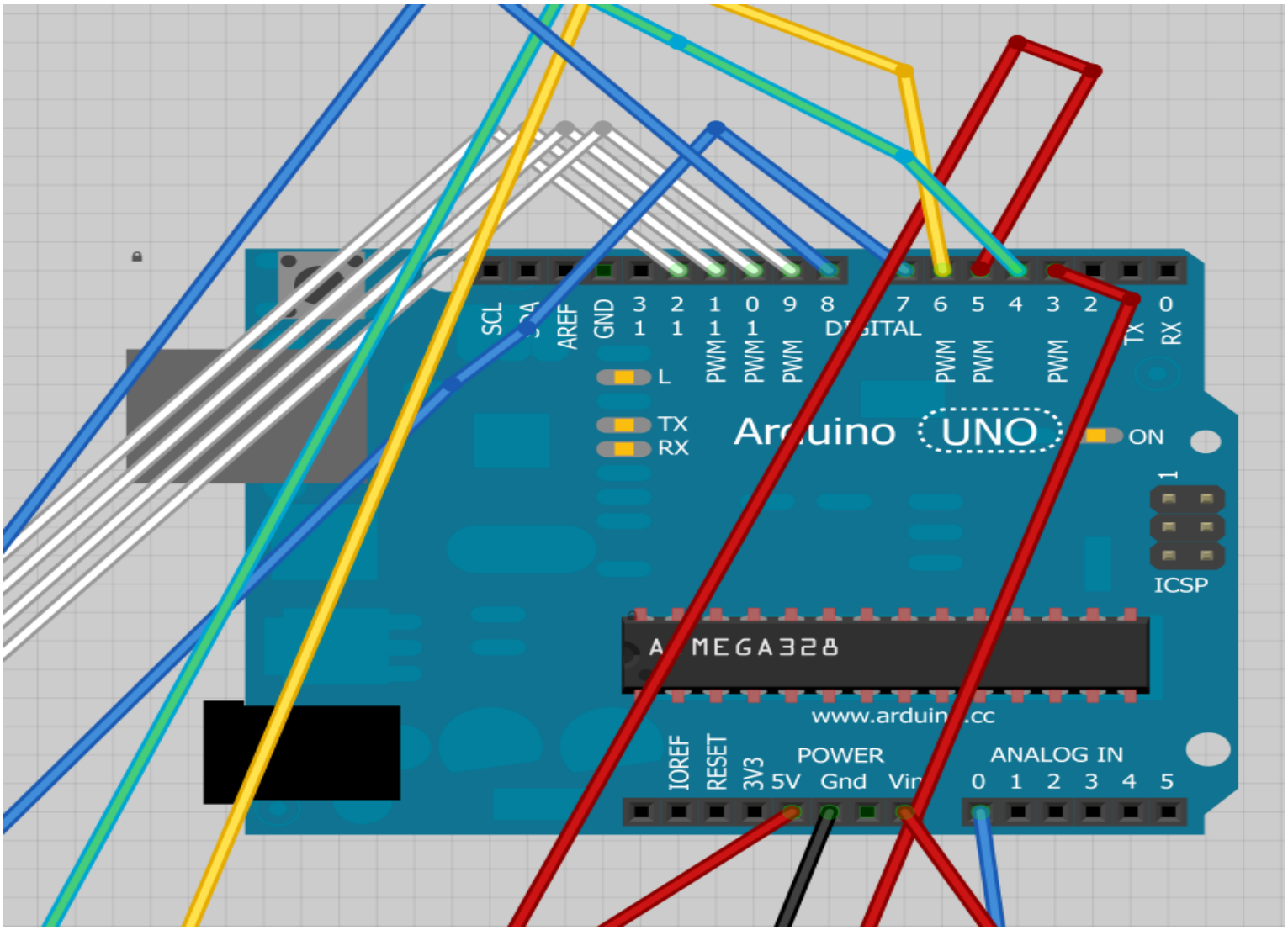
-Regulador de voltaje de 9V 7809.

- **2-Circuito armado.**

A continuación, detallamos el esquema del circuito final que armamos para nuestro proyecto. Y en el desarrollo del informe explicaremos cada parte del mismo:



Zoom en Arduino y sus conexiones (Igualmente en el informe están explicadas):



- # 3- Explicación de cada parte.

3.1-Fuente lineal

¿Para qué la necesitamos?: Necesitamos una fuente lineal para poder alimentar con 9 V DC al arduino y al ventilador, conectando la misma directamente a la toma de corriente de 220V CA.

Esta fuente, está formada por:

-Un "soporte de seguridad" **(3.1.1)** que encontramos; que tiene 3 funciones principales: 1-Controlar que la corriente que circule no sea mayor a la esperada (mediante el fusible y el inductor). 2-Controlar el flujo de esta corriente (si pasa o no al circuito, mediante la llave). 3-Que a la salida de este "soporte" se obtengan 9V CA eficaces (gracias al transformador).

-Un Circuito Impreso **(3.1.2)** en el cual llega a la entrada los nombrados anteriormente 9V CA eficaces (aunque realmente lo medimos y eran 9,9V CA eficaces, NO 9V como indicaba en las especificaciones del propio transformador); y salen 9V CC (8,93V CC medidos para ser exactos).

3.1.1-Soporte de seguridad

Este "soporte", lo encontramos dentro de los materiales reciclables dentro del aula. Contenía componentes eléctricos que nos servía, como por ejemplo la entrada (conector hembra) para un cable de 220V CA, un interruptor o llave (1-encendido, 0-apagado), un inductor y un fusible. Utilizamos esta estructura para conectarlo a nuestro transformador. Para conectarlo, empalmamos los cables del transformador con la salida de la llave (cable azul) y la salida del inductor (cable marrón) como se muestra en la ***imagen 1.1*** (marcados en rojo).

Imagen 1.1

Esquema de este circuito, vista frontal (***imagen 1.2***) y trasera **(*imagen 1.3*):**

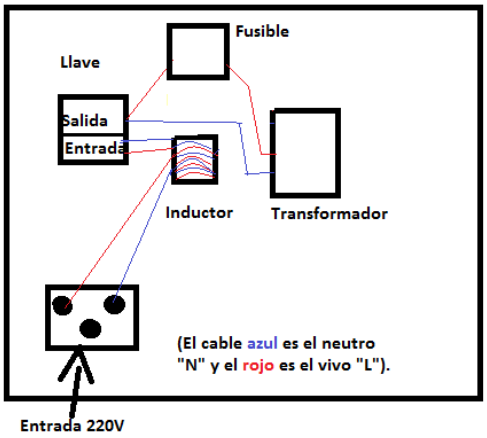


Imagen 1.2

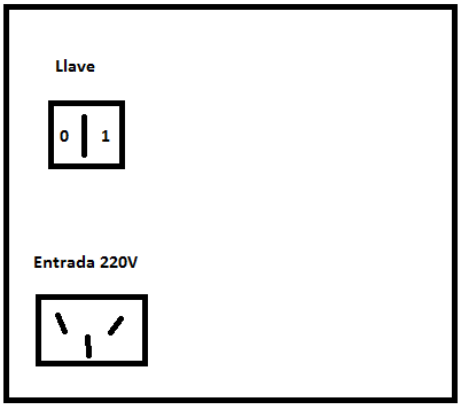
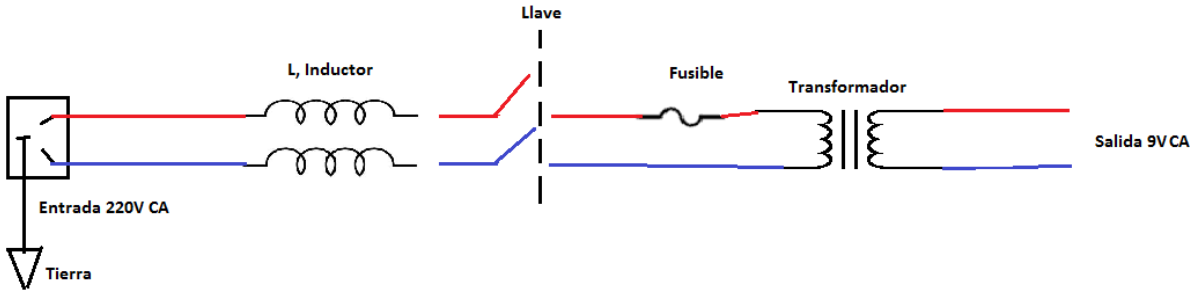


Imagen 1.3

Comportamiento y función de cada componente:



La llave sirve para dejar pasar o no la corriente y así hacer funcionar o no al circuito entero. Cuando la llave está abierta (posición 0), no hay corriente en el circuito, cuando la llave está cerrada (en la posición I), deja pasar la corriente.

El fusible es un componente eléctrico hecho de un material conductor, generalmente estaño, que colocamos para interrumpir la corriente cuando ésta es excesiva (ya que se derrite a una determinada corriente, actuando así como un mecanismo de seguridad de toda la instalación).

Un inductor o bobina es un componente pasivo de un circuito eléctrico que, debido al fenómeno de la autoinducción, almacena energía en forma de campo magnético. En nuestro caso, lo utilizamos para filtrar los picos de corriente; ya que cuando hay bajas frecuencias actúa como un cable y a altas frecuencias como circuito abierto. Las frecuencias bajas se pueden filtrar mediante un filtro pasa alto, ya que este mismo solo deja pasar frecuencias superiores a un cierto valor específico, filtrando las frecuencias menores.

El transformador es un dispositivo eléctrico que permite aumentar o disminuir la tensión en un circuito de corriente alterna. Nosotros lo utilizamos para que a la entrada de este le lleguen los 220V CA eficaces (311 V pico) y a la salida obtengamos 9V CA eficaces ó 12,7 V pico (pero que realmente salen 9,9V CA eficaces, 14V pico; los cuales medimos).

3.1.2-Circuito impreso

Está ubicado a la salida del **3.1.1**; le llegan 9,9V CA eficaces (14V pico) y salen los 9V DC (8,93V DC medidos para ser exactos) que necesitamos para conectarlo al Arduino y al ventilador.

Junto con el transformador del **3.1.1** forman el llamado "circuito rectificador de onda completa"; el cual está formado por 4 diodos, o también llamado "puente de diodos" (también se podían hacer con 2 diodos pero necesitábamos que el transformador tenga un "punto/derivación media" y no lo tenía). Este circuito, como lo dice su nombre RECTIFICA el voltaje de CA, para así convertir una señal de valor promedio=0 a una de valor promedio distinto de 0. De esta forma, observamos la salida del transformador (***Imagen 1.4***) y la salida tras rectificarse la señal (***Imagen 1.5***):

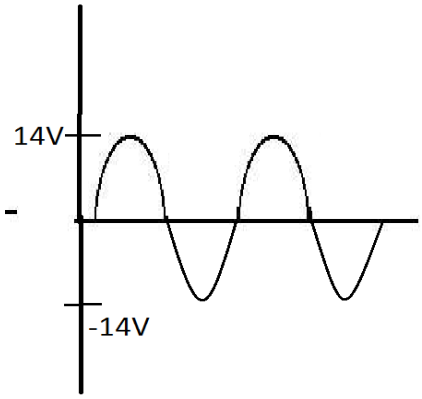


Imagen 1.4

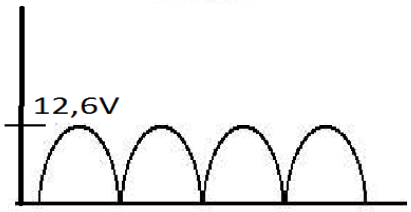


Imagen 1.5

A la salida de estos diodos hay 2 capacitores en paralelo: Uno de 1000uF para filtrar los ruidos de BAJAS frecuencias y para disminuir o reducir notablemente el **rizo** (explicado más detalladamente en la sección 4, ver índice); y otro de 0.1uf para filtrar los ruidos de ALTA frecuencia.

Por último, a la salida de estos capacitores, conectamos un regulador de tensión fija (LM7809) que regulan los 12.6V CA anteriores a 9VCC. El LM7809 tiene como especificaciones que le llegue como mínimo 11V para regular bien (Ver ***imagen 1.6***). Y como entran 12,6V entonces funciona correctamente. Para su conexionado, observamos la ***imagen 1.7.***

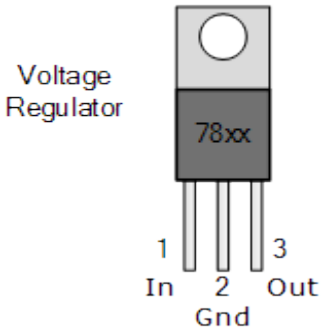


Imagen 1.7

Type	Min Input Voltage	Output Voltage
7805	7V	+5V
7806	8V	+6V
7808	10V	+8V
7809	11V	+9V
7812	15V	+12V
7815	18V	+15V
7818	22V	+18V
7824	30V	+24V

Imagen 1.6

De esta forma, se obtienen de una corriente de 220VCA, 9VCC.

El circuito que utilizamos está detallado en la ***Imagen 1.8***.

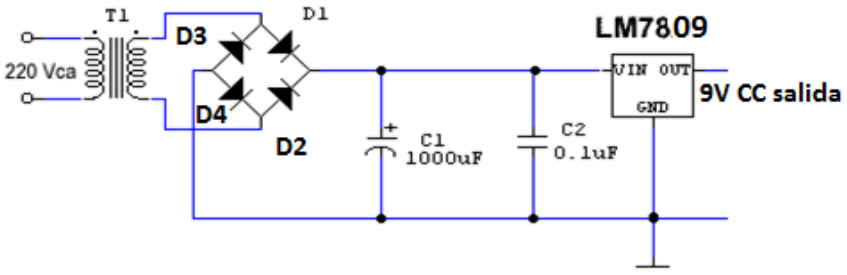


Imagen 1.8

Todo este circuito (a excepción del transformador) lo montamos en un **Circuito Impreso**. Este tiene 2 "partes", una base formada por el PCB (parte de Pertinax que no conduce, donde coloco los componentes electrónicos) y una parte de cobre (parte conductora en la cual "formamos" las pistas de conexión que nosotros necesitamos oxidando las partes que no queremos).

Pasos que seguimos para hacer el Circuito Impreso:

Paso 1: Primero esquematizamos como sería nuestro circuito en la placa (***Imagen 1.9***).

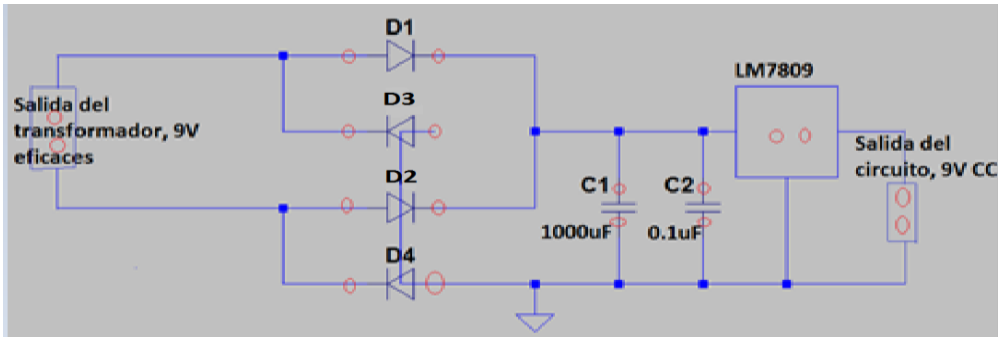


Imagen 1.9

(De esta manera la salida + del transformador entra al ánodo del D1 y al cátodo del D3 y la salida – del transformador entra al ánodo del D2 y al cátodo del D4).

*Aclaración: los círculos rojos representan donde tuvimos que perforar luego (Paso 4) para colocar los componentes.

Luego, dibujamos en la base de la placa (la de Pertinax) nuestro circuito como se ve a en la ***Imagen 2***. (Este paso no era necesario, ya que se podía dibujar directamente en la parte de cobre pero espejado).

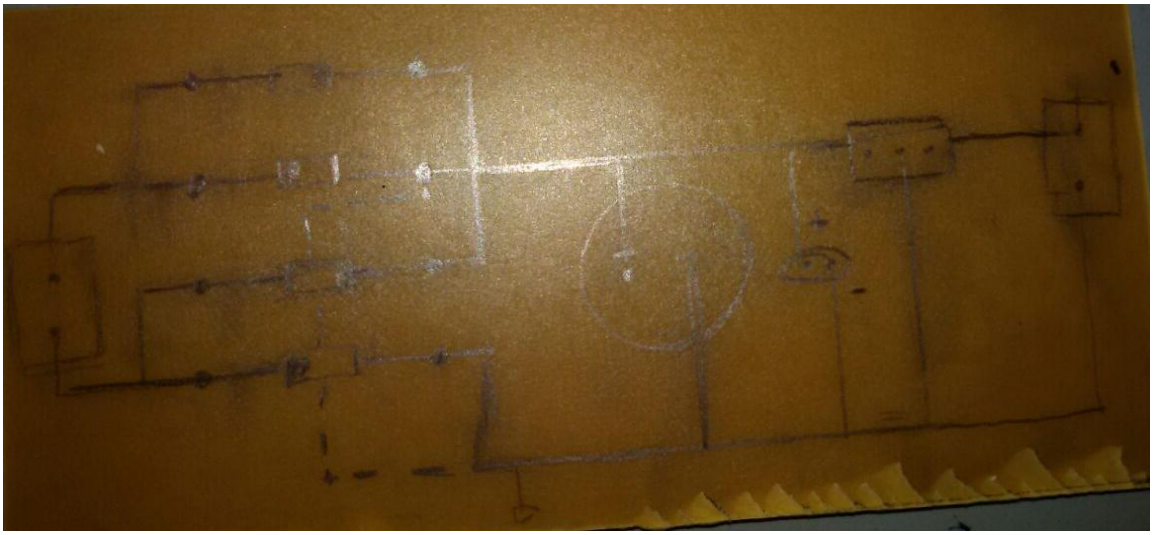


Imagen 2

Paso 2: Redibujamos el circuito en el cobre (el material conductor) comparándolo con la base de Pertinax para que queden "en espejo" (***Imagen 2.1***). Y luego pintamos las pistas que nosotros queríamos con fibrón negro indeleble (***Imagen 2.2***; donde la parte de abajo es tierra, la anchura de esta pista depende de la cantidad de corriente que circula, generalmente se hace ancha y decidimos hacerla así).

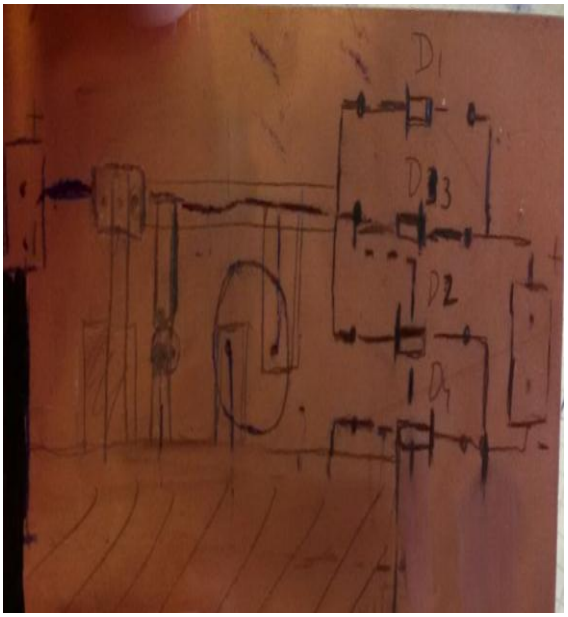


Imagen 2.1

Imagen 2.2

Paso 3: Colocamos Ácido (Cloruro Férrico Concentrado, ***Imagen 2.3***), en un recipiente hasta formar una "capa" de este; y luego colocamos la parte del circuito de cobre en el ácido (***Imagen 2.4)*** para que el mismo lo "coma", menos las partes que habíamos marcado anteriormente con fibrón indeleble (que mantiene el cobre aislado del ácido); así se forma la pista que nosotros necesitamos.

Imagen 2.3

Imagen 2.4

Paso 4: Hicimos los agujeros correspondientes según donde iban los componentes mediante un taladro con una mecha fina especial para hacer circuitos impresos.

Paso 5: Colocamos los componentes en sus respectivos lugares en la base de la placa (***Imagen 2.5***).

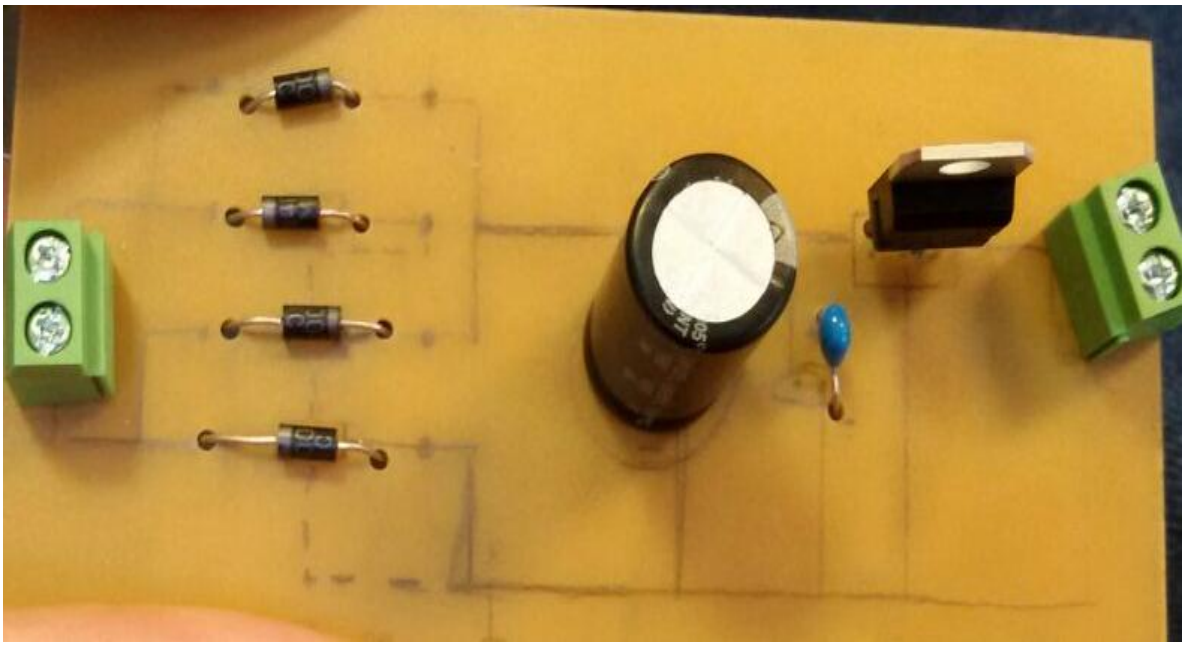


Imagen 2.5

Paso final: Soldamos con estaño para que los componentes queden fijos a la placa (en la ***Imagen 2.6*** observamos los componentes SIN soldar, y en la ***Imagen 2.7*** ya están soldados).

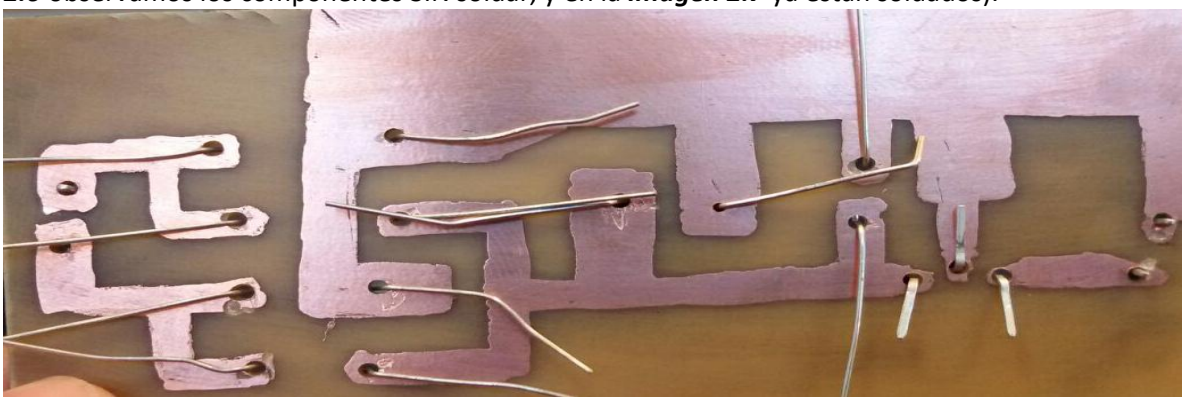


Imagen 2.6

Imagen 2.7

Al tenerlo ya terminado al Circuito Impreso, conectamos la entrada de este circuito (3.1.2) a la salida del transformador (salida del circuito 3.1.1) como se muestra en la ***imagen 2.8*** (marcado en rojo).

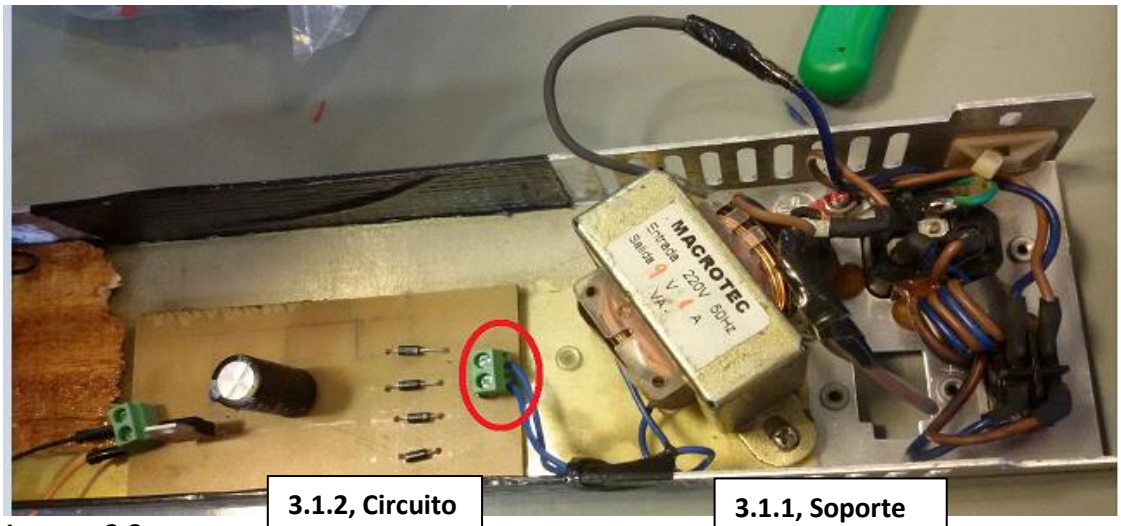
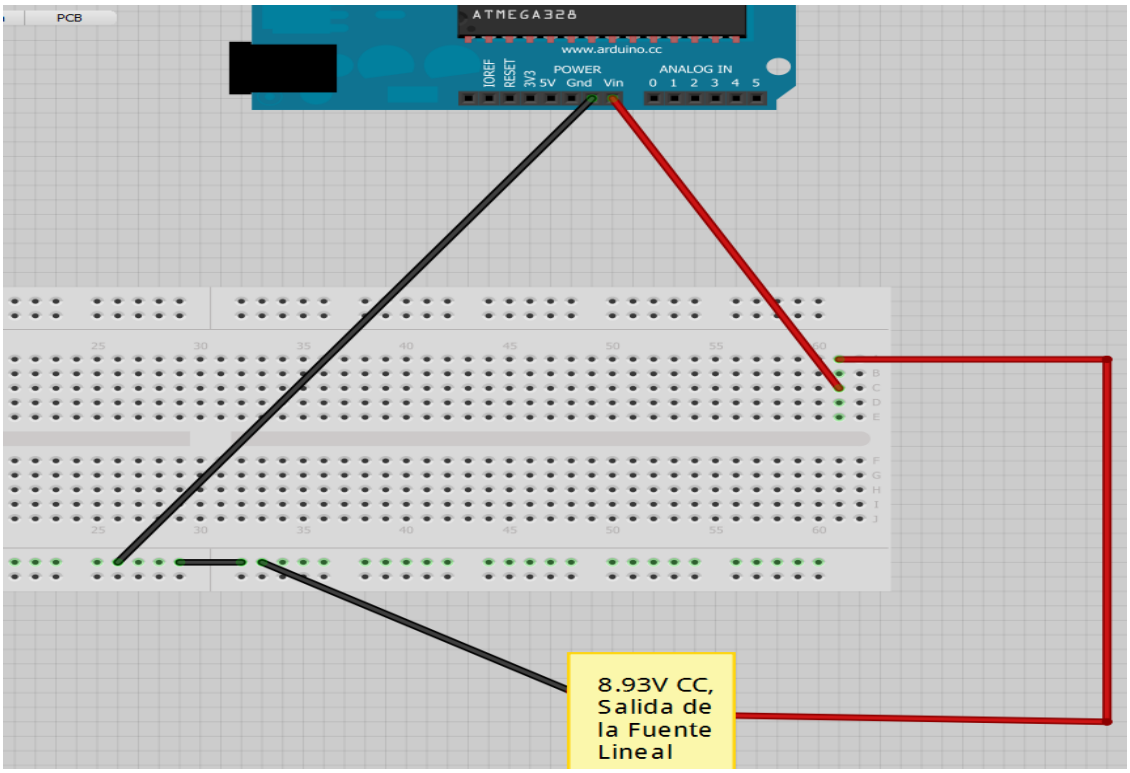


Imagen 2.8

De esta manera, teniendo ya los 8,93V alimentamos directamente a Arduino conectando la salida + del Circuito impreso al protoboard y luego este al pin Vin de Arduino; y conectando la salida – a GND en el protoboard (el cual está en la misma línea que pusimos GND de Arduino). Esto está detallado en el ***Circuito 1.***



Circuito 1

▪ 3.2-Arduino

¿Para qué lo necesitamos?: Para programar el comportamiento de los leds, y controlar la velocidad del ventilador, como así para mostrar la temperatura que emite el sensor por el display LCD. El Arduino que utilizamos es el "Arduino Uno" con un microcontrolador ATmega328P. Arduino es una plataforma de hardware y software de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación Processing. Es decir, una plataforma de código abierto para prototipos electrónicos. El microcontrolador (µC) dentro del Arduino, es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Primero observamos el datasheet de Arduino Uno (***Imagen 2.9***), en el cual marcamos con rojo las principales características que tuvimos en cuenta.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Imagen 2.9

Así, sabemos que por cada pin circulan como máximo 20mA y que la entrada al Arduino por el pin Vin es de 7-12V (en nuestro caso son 8,93V los Volts que entran, lo cual está dentro de estos parámetros). Además, observamos que opera con 5V (o sea que cada pin entrega 5V). En la ***Imagen 3***, se detalla un esquema general de la placa Arduino Uno.

Imagen 3

Explicación pines Arduino UNO:

Entradas y salidas digitales: Están situadas en la parte de arriba de la placa, van del 0 hasta el 13, este último pin lleva una resistencia interna incluida. La señal digital puede estar o encendida o apagada (LOW o HIGH). Los pines cero y uno se pueden utilizar para cargar el programa en la placa.

Salidas analógicas: Son los pines 11, 10, 9, 6, 5 y 3. Son salidas PWM (Pulse Width Modulation) que realmente son salidas digitales que imitan salidas analógicas, modificando la separación entre los diferentes pulsos de la señal. La señal PWM puede dar diversos valores hasta 255, se utilizan, por ejemplo para variar la intensidad de un LED o hacer funcionar un servo (en nuestro caso lo utilizamos para el ventilador que gire de acuerdo a un determinado ciclo de trabajo). Hay que decir que estos pines funcionan como salidas o entradas digitales o como salidas analógicas.

Entradas analógicas: Son los pines A0, A1, A2, A3, A4 y A5 (Analog in). Se utilizan para que entre una señal de un sensor analógico de valor variable (en nuestro caso el sensor de temperatura). También se pueden utilizar como pines digitales.

Pines y conectores de alimentación:

- Pin GND: Son los pines a tierra de la placa, el negativo.
- Pin 5v: Por este pin suministra 5v
- Pin 3,3v: Por este pin suministra 3,3v
- Pin Vin: Voltaje de entrada, por este pin también se puede alimentar la placa.
- Pin RESET: Por este pin se puede reiniciar la placa
- Pin IOREF: Sirve para que la placa reconozca el tipo de alimentación que requieren los shields
- Pin AREF: Sirve para suministrar un voltaje diferente a 5v por los pines digitales.
- Conector USB: para cargar el programa.
- Conector de alimentación: para alimentar a la placa (en nuestro caso utilizamos el pin Vin para alimentarla).

Arduino, cuenta sólo con un sistema de **conversión analógico a digital, CAD (*Imagen 3.1*)**; el cual lo explicaremos a continuación. Estos CAD, se encuentran en los pines A0, A1, A2, A3, A4 y A5, y son utilizados para leer por ejemplo valores de sensores ó potenciómetros (en nuestro caso lo usamos para leer los valores del sensor LM35 con una señal es amplificada por el CA3140EZ). Arduino **NO posee un sistema de conversión digital a analógico, CDA.**

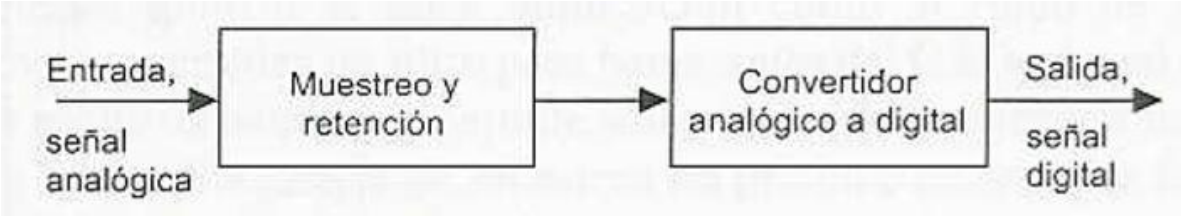


Imagen 3.1

Un CAD 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) → ***Imagen 3.2***.

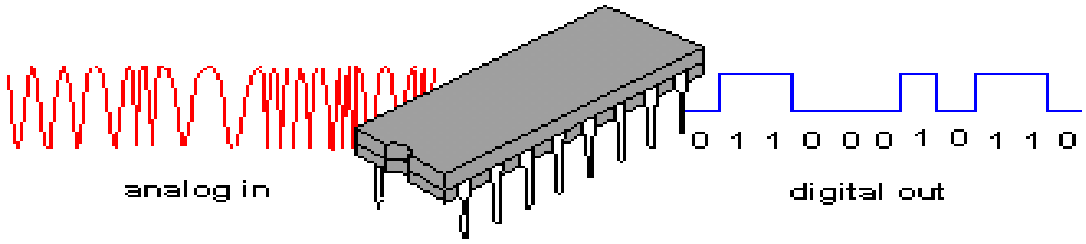


Imagen 3.2

<u>El proceso de conversión A-D consiste en 4 pasos:</u>

1- Muestreo: donde se toman muestras periódicas de la amplitud (valor pico) de la señal analógica. (***Imagen 3.3***).
2- Retención: las muestras tomadas son retenidas por un circuito de retención (hold), el tiempo suficiente para medir su nivel (cuantificación).
3-Cuantificación: donde se mide el nivel de voltaje de <u>cada una</u> de las muestras. ***(Imagen 3.4).***
4-Codificación: se traducen los valores obtenidos durante la cuantificación al código binario. (***Imagen 3.5***).

Imagen 3.3

Imagen 3.4

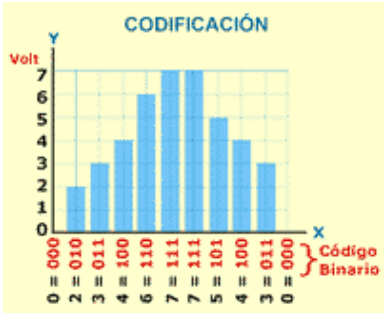


Imagen 3.5

La distancia entre los muestreos tomados ("cuadrados" azules de la ***Imagen 3.5***), es la **Resolución**. **Arduino**, establece una relación entre su entrada (señal analógica) y su salida

(señal digital) dependiendo de esta resolución, la cual repercute en la precisión con la que se reproduce la señal original. ***Resolución = +Vref/2^n*** (donde *n* son bits de resolución)

Arduino utiliza un conversor A/D de 10 bits (convierte valores continuos entre 0 y 5V a valores enteros entre 0 y 1023). Si hago 2^n (donde n son los bits de resolución), obtengo la cantidad total de valores que el Arduino puede tomar en la lectura; como n=10 y 2^{10}=1024; de 0-1023 son la cantidad de valores totales que nos muestra Arduino en la lectura.

Para leer una entrada analógica (en nuestro caso la del LM35, el cual conectamos su salida de lectura al pin A0), usamos la función analogRead() del IDE de Arduino, que asigna a tensiones entre 0 y 5V valores enteros entre 0 y 1023, proporcionando una resolución de lectura de 4,8mV por unidad (cada "cuadrado" azul de la ***Imagen 3.5*** tiene 4,8mV de "largo"). Estos 4,8mV los saco de la ecuación ya antes vista: Resolución = Vref/1024; donde Vref=5V. El rango de entrada y la resolución se pueden cambiar usando la función analogReference() pero en nuestro caso no la usamos.

Por esto en el código que utilizamos para medir la temperatura (Ver ***Apéndice***) primero leímos los datos del sensor mediante *analogRead(Sensor*); y luego transformamos esta lectura analógica de 0-1023 a un Voltaje de 0-5V.

▪ 3.3-Sensor de temperatura y amplificador operacional no inversor.

El **sensor de temperatura** que usamos fue el LM35, ya que su respuesta es bastante lineal (Ver ***Imagen 3.8***). Viendo su datasheet, observamos las "patas" de este para su conexionado (***Imagen 3.6***); y además sus especificaciones (***Imagen 3.7***). En sus especificaciones, observamos que puede alcanzar a medir temperaturas entre 55°C y 150°C, y que su salida analógica, mide 10mV/°C (de esta forma, 270mV son equivalentes a 27°C).

SENSOR DE TEMPERATURA - LM35

Imagen 3.6

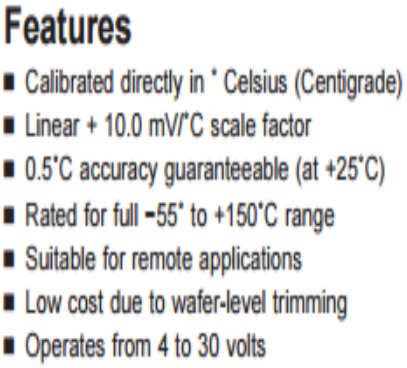


Imagen 3.7

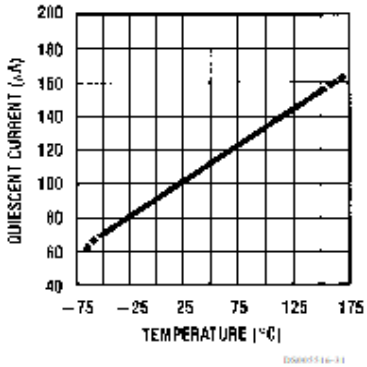


Imagen 3.8

Para probar el sensor, utilizamos una resistencia de potencia, la cual al conectarla a una fuente la hacemos que se "caliente", que disipe potencia. Como máximo disipa 5W de potencia. La resistencia de potencia que utilizamos eran 2 resistencias de 47ohms en paralelo (Resistencia equivalente de 25,5ohm). Para probar al sensor con esta resistencia, necesitábamos saber el valor de la fuente a la cual debíamos conectarla; y mediante cálculos la hayamos:

Si queríamos que disipe la potencia máxima (5Watts), necesito una fuente de V de… → $P=V^2 / R$ → $5W=V^2 / 25,5ohm$ → **V= 11,3V.** Y si queríamos una potencia "media" (2W), **V=7,14V**. Sacamos la corriente para este voltaje (sólo para saber); la cual es de → V/R=I → 7,14V/25,5ohm= I → **I=0,28A.**

Decidimos conectar la resistencia de potencia a una fuente con un valor de **10,4V (*Imagen 3.9*.)** (de esta forma la potencia que suministra es → $P= (10,4V)^2/25,5ohm$ → **P=4,24W).**

Luego, armamos el circuito con el sensor LM35 (viendo las patas indicadas en **la *Imagen 3.6***: GND (tierra del sensor) → al GND del arduino; VCC (alimentación del sensor) →a los 5V del arduino; y V OUT (salida analógica del sensor) →De esta pata sacamos un cable para medir con el voltímetro cuantos volts salen). En las ***imágenes 4 y 4.1*** medimos esto (la punta negra del voltímetro lo colocamos a la tierra del protoboard y la roja a la pata del medio, OUT), acercando la resistencia de potencia al LM35 y viendo como variaba el voltaje medido.
De esta manera, medimos 160mv (o sea 16°C) en un principio, y luego, al acercarle la resistencia de potencia, 260mV (o sea 26°C). Tuvimos un pequeño problema para medir estos valores muy chicos de V, ya que había que conectarlo bien las salidas del voltímetro al cable de tierra y al que salía de la pata del medio con **cocodrilos** porque mis manos no permitían medir estos mV.

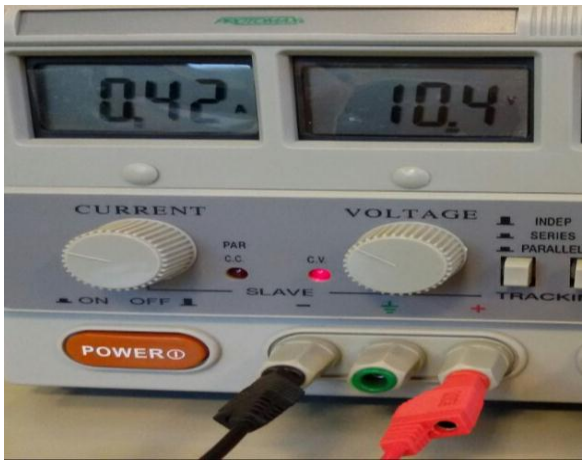


Imagen 3.9

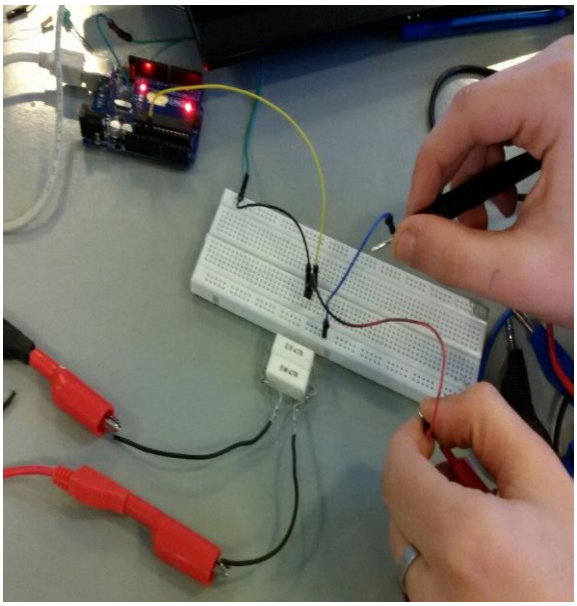


Imagen 4

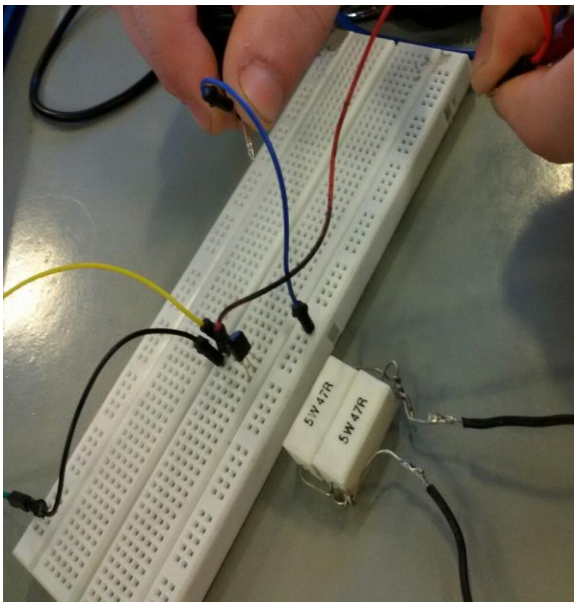


Imagen 4.1

Para solucionar el de que los V medidos son en una escala muy chica, mV; usamos **un amplificador operacional no inversor**. De esta manera, conectamos a la entrada de este la salida analógica del LM35 (patita del medio); así, al entrar los mV medidos, se amplifican al rango de los V y así la señal sale amplificada y NO invertida. Este amplificador además debe ser realimentado con los VCC correspondientes. El VCC+ es de 5V (lo conectamos directamente a Arduino) y el VCC- es 0V (ya que nuestro rango de temperatura no va a llegar a las mediciones negativas, lo conectamos a GND). De esta manera, **nuestro rango de temperaturas que podemos llegar a medir, va a ser desde los 0mv (0°C) hasta aproximadamente los 500mV (50°C).**

Utilizamos el amplificador no inversor **CA3140EZ**, al cual observamos su conexionado viendo su datasheet (***Imagen 4.2***).

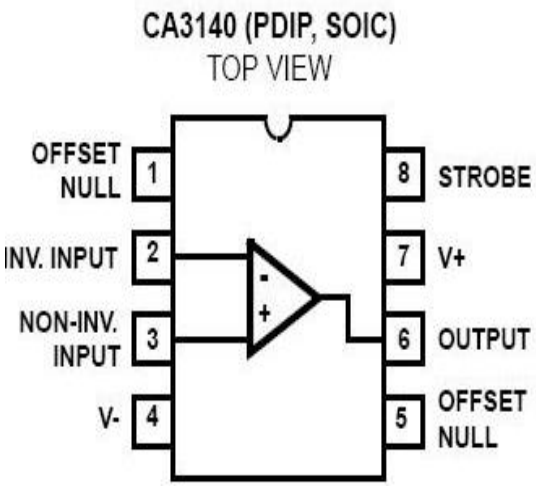


Imagen 4.2

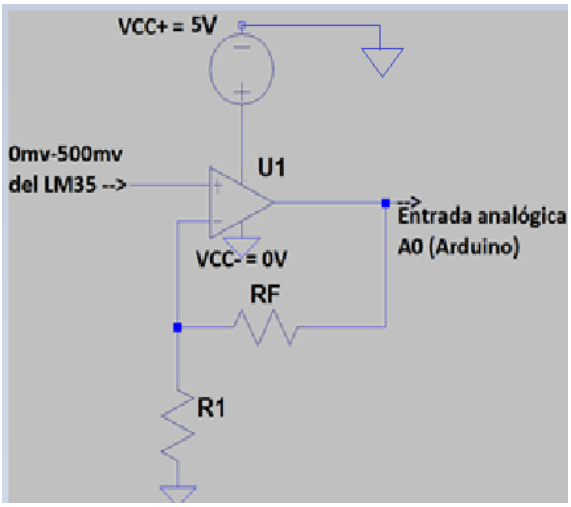


Imagen 4.3

Luego, viendo la configuración del AMP OP no inversor (***Imagen 4.3***), y sabiendo que su relación de ganancia es: **Av=1 + (Rf/R1)**; calculamos las resistencias R1 y RF utilizadas, así como su ganancia.

Primero calculamos con una Av=9 y una R1=560ohm, que RF utilizamos:

9= 1 + (Rf/560ohm) → 8=Rf/560ohm → Rf= 4480ohm.

(Aclaración= utilizamos una Av=9 ya que antes habíamos planteado otras ganancias pero nos daban valores de resistencias que no teníamos, entonces con una ganancia de 9 nos dio resistencias con valores que sí teníamos).

Uso una Rf de valor parecido, de 4,7Kohm; y ahora calculo la ganancia real que tendría con estas 2 resistencias:

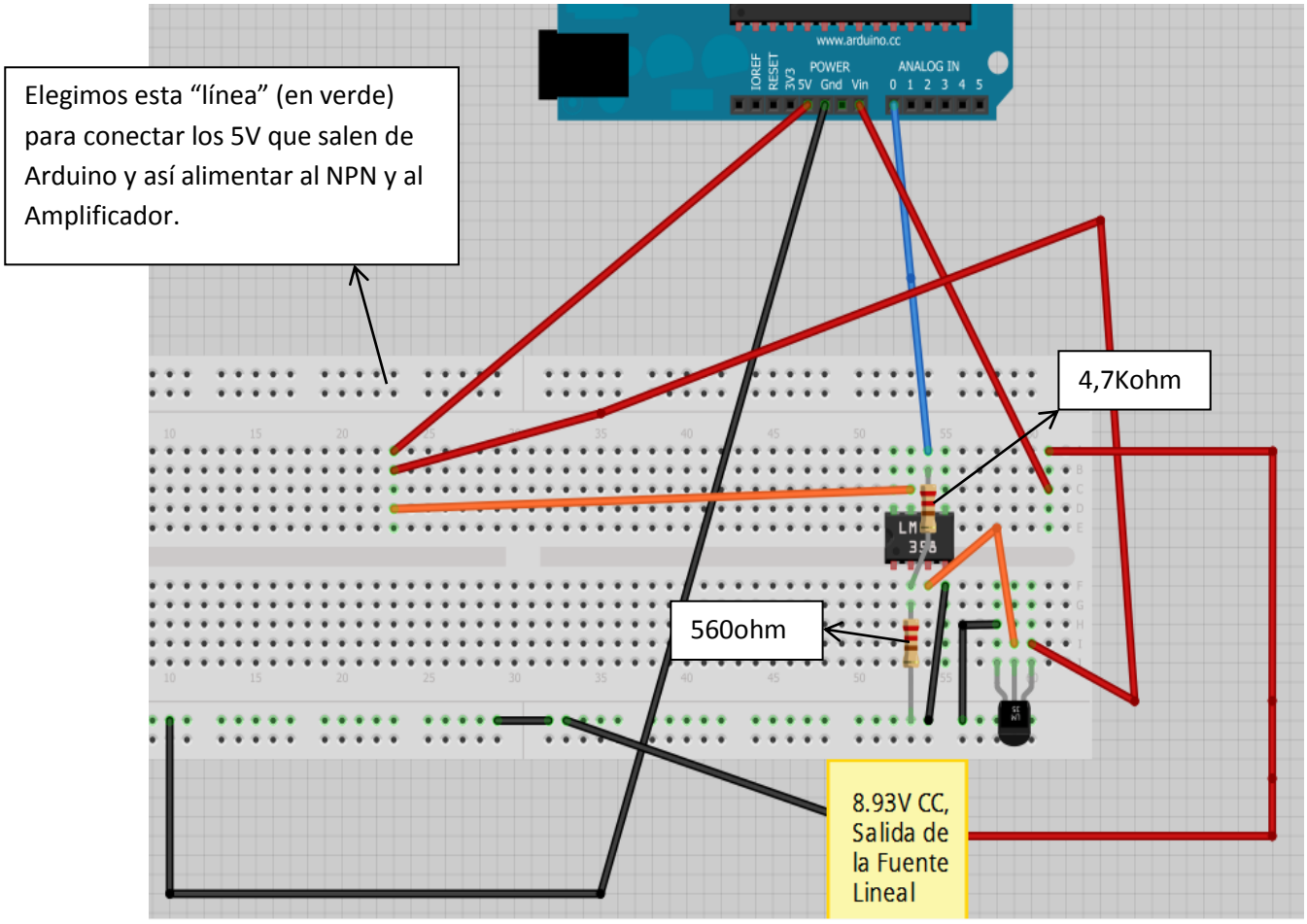
Av=1+(4700ohm/560ohm) → **Av=9,4**

Esto quiere decir si se ingresan por ejemplo 180mv (indica temperatura de 18°C) por la entrada + del amplificador, entonces a la salida se obtendrán (180mV * 9,4) → 1,7V.

Pero midiendo con el voltímetro a la salida del amplificador (dados estos 180mV de entrada) me indica que son 1,76V; dada esta medición concluyo que la ganancia del amplificador no inversor es de aproximadamente → **Av=9,7**. En conclusión, las resistencias que utilizo son de → **Rf=4,7Kohm y R1=560ohm.**

De esta manera, si tengo por ejemplo una temperatura de 19.9°C, indica que a la salida del sensor LM35 hay 199mV (ya que son 10mV por cada 1°C). Y a la salida del amplificador deberían haber → Vi. Av=Vo → 199mV.**9,7**=Vo → **Vo=1,93V**. Esto se cumple.

De esta manera conectamos el Sensor de Temperatura y el Amplificador como indica el ***Circuito 2.***



Circuito 2.

3.4-Transistor NPN

¿Para qué lo necesitamos?: Lo necesitamos, para que, dado el voltaje de entrada (proveniente del pin 3 de arduino, el cual suministra 5V) que llega a la BASE del NPN (con un DC determinado), "llegue" al ventilador un Voltaje VRC de 8,8V **con el mismo DC.**

(Aclaración: son 8,8V por la ecuación →9V-VRC-VCE(sat.)=0V →Supongo que VCE(sat.)=0,2V → 9V-VRC-0,2V=0 → VRC=8,8V)

De esta forma si entran 5V con **DC=100%,** en VRC (V del ventilador) tengo 8,8V con **DC=100%.** Realmente son **8,6V** los que llegan al ventilador, ya que medimos que VCE(sat.)=0,4V y NO 0,2V como supusimos anteriormente.

Así, logramos lo que queríamos, **que a una temperatura determinada se entreguen al ventilador una misma cantidad de volts (8,6V), pero distinto ciclo de trabajo, variando así su velocidad.**

Para que pase esto el transistor debe estar **saturado**. El transistor bipolar NPN, actúa en corte o en saturación. Si esta en corte la corriente de la base (IB) es 0A, por lo tanto la corriente del colector y emisor es también 0A, no conduce; además las uniones B-E y B-C están polarizadas en inversa. Al aumentar IB, IC también aumenta y VCE disminuye como consecuencia del mayor V que cae en RC (RC en nuestro caso es el ventilador).

Si trabaja en **saturación** IC va a ser = a VCC/RC (ya que VCE=0 aproximadamente; al medirlo luego de armar el circuito nos dio 0,4V, lo cual es correcto). En este modo de funcionamiento el transistor no ejerce ningún control sobre la corriente de salida y funciona como una llave CERRADA. Así, el circuito que queremos inicialmente es el ilustrado en la ***Imagen 4.4*** (Donde RC es el ventilador, y R1 es la resistencia que calculamos más abajo para la Base).

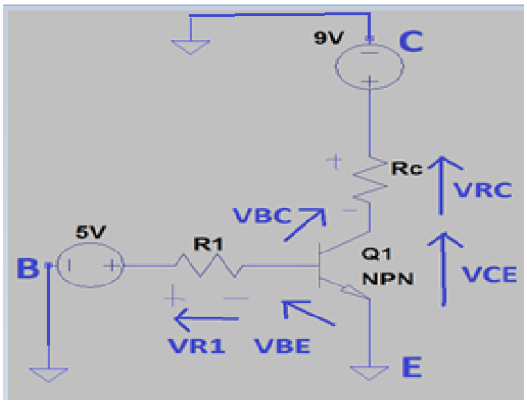


Imagen 4.4.

La corriente principal (IC) entra por el colector C y sale por el emisor E. Esta IC es la I que sale de la fuente lineal de 9V. Medimos la IC conectando a la fuente el ventilador directamente y nos dio aprox. 0,17mA (***Imágenes 4.5 y 4.6***).

Imagen 4.5

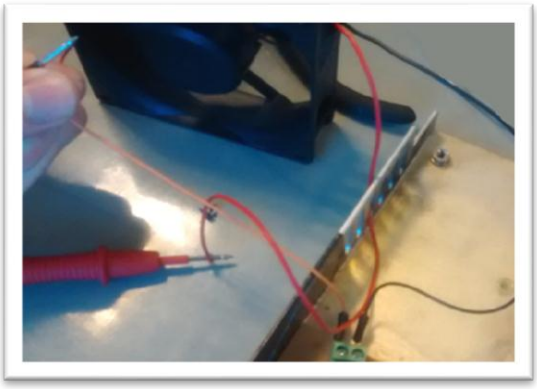


Imagen 4.6

(Aclaración: dio negativo ya que colocamos las puntas en el sentido contrario, por esto nos dio que la corriente circulaba en este sentido; entonces la corriente I es de 0.166mA, aprox. 0.17mA)

Pero antes de utilizar el circuito con el NPN, debemos ver qué valor de voltaje y corriente soporta el motor (marcados en rojo en el datasheet del Ventilador/ Motor trifàsico - **KD1209PTS1 H;** ***Imagen 4.7***):

SUNON

SPECIFICATIONS

MODEL : KD1209PTS1
P/N : H

1-1. Rated Voltage	:	12 VDC
1-2. Operating Voltage Range	:	5-13.8 VDC
1-3. Starting Voltage	:	5 VDC
1-4. Rated Speed	:	2500 RPM ± 300 RPM
1-5. Air Delivery	:	45.5 CFM
1-6. Static Pressure	:	0.14 Inch-H_2O
1-7. Rated Current	:	0.22 AMP / 0.25 AMP MAX
1-8. Rated Power	:	2.6 WATTS / 3.0 WATTS MAX

Imagen 4.7

Dados estos valores, los 170ma ni se acercan a los 0,25A máximos; entonces continuamos.

Ahora vamos a tener en cuenta qué transistor usar, ya que necesitamos uno que soporte 170mA, en este caso el que usamos fue el TRANSISTOR **BD437**.

En el datasheet del NPN (***Imagen 4.8)*** se indica que pueden circular como máximo 4 A (entonces soporta 170mA).

MAXIMUM RATINGS

Rating		Symbol	Value	Unit
Collector–Emitter Voltage	BD437 BD441	V_{CEO}	45 80	Vdc
Collector–Base Voltage	BD437 BD441	V_{CBO}	45 80	Vdc
Emitter–Base Voltage		V_{EBO}	5.0	Vdc
Collector Current		I_C	4.0	Adc
Base Current		I_B	1.0	Adc
Total Device Dissipation @ T_C = 25°C Derate above 25°C		P_D	36 288	Watts W/°C
Operating and Storage Junction Temperature Range		T_J, T_{stg}	–55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Case	θ_{JC}	3.5	°C/W

Imagen 4.8

Luego, nos fijamos que curva de IB e IC en función de VCE (en saturación) se "adapta" a nuestra corriente de 170mA (es más equivalente) → ***Imagen 4.9***; por esto vemos que la de 100ma se adapta más.

BD437 BD441

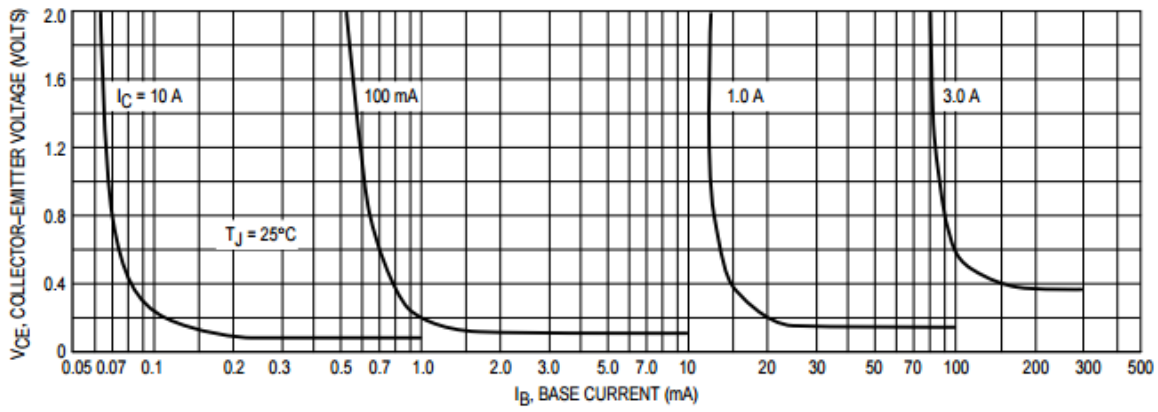


Imagen 4.9

En dicha curva, los valores desde 2 a 10mA la corriente IB es constante, así elijo arbitrariamente un valor entre estos 2 valores de corriente, **5mA.**

De esta manera para que el transistor esté saturado, IB=5mA e IC= 100mA. Así calculo el Beta forzado / ganancia de corriente forzada (Bf):

Bf= IC (sat.) / IB (sat.) → Bf = 100ma / **5ma** → Bf=20.

El cálculo anterior lo hicimos para el caso de una IB de 5mA e IC de 100mA; pero en nuestro caso el único dato que tenemos es IC=170mA. De esta manera, suponemos un valor de BF de 20, y así nuestra IC va a ser 20 veces mayor a IB.

Así, volviendo a nuestro circuito (***Imagen 4.4***); y sabiendo que VCE(sat.)=0,2V y VBE=0,7V; con el valor de Bf (20) e IC (170mA), calculamos nuestra IB:

Bf= IC(sat.)/IB(sat.) → IB(sat.)=IC(sat.)/Bf → IB= 170mA / 20 → **IB=8,5mA**

Sabiendo IB, ahora calculo la Resistencia R1 necesaria analizando la maya de base:

5V-VR1-VBE (sat.)=0V → 5V - VR1 – 0,7V =0 → 4,3V – 8,5mA . R1 = 0 → **R1= 505ohm**

Con esta resistencia, IC va a ser 20 veces mayor a IB; entonces colocamos una **R1 de 470ohm** (aprox. 505ohm).

De esta manera conectamos el NPN como indica el ***Circuito 3***; viendo la ubicación de cada pata (***Imagen 5***), y con una R1=470ohm. Así, la patita del C conectamos al negativo del ventilador (cable negro, pero es el amarillo en el ***Circuito 3***). El positivo del ventilador (cable rojo) directamente a 9V. La patita B la conectamos a los 5V de arduino (pin 3 con PWM) y la patita E a GND.

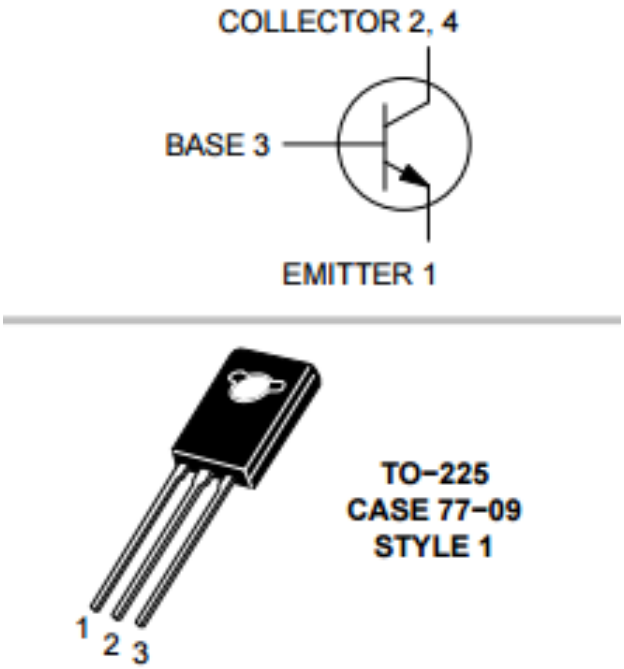
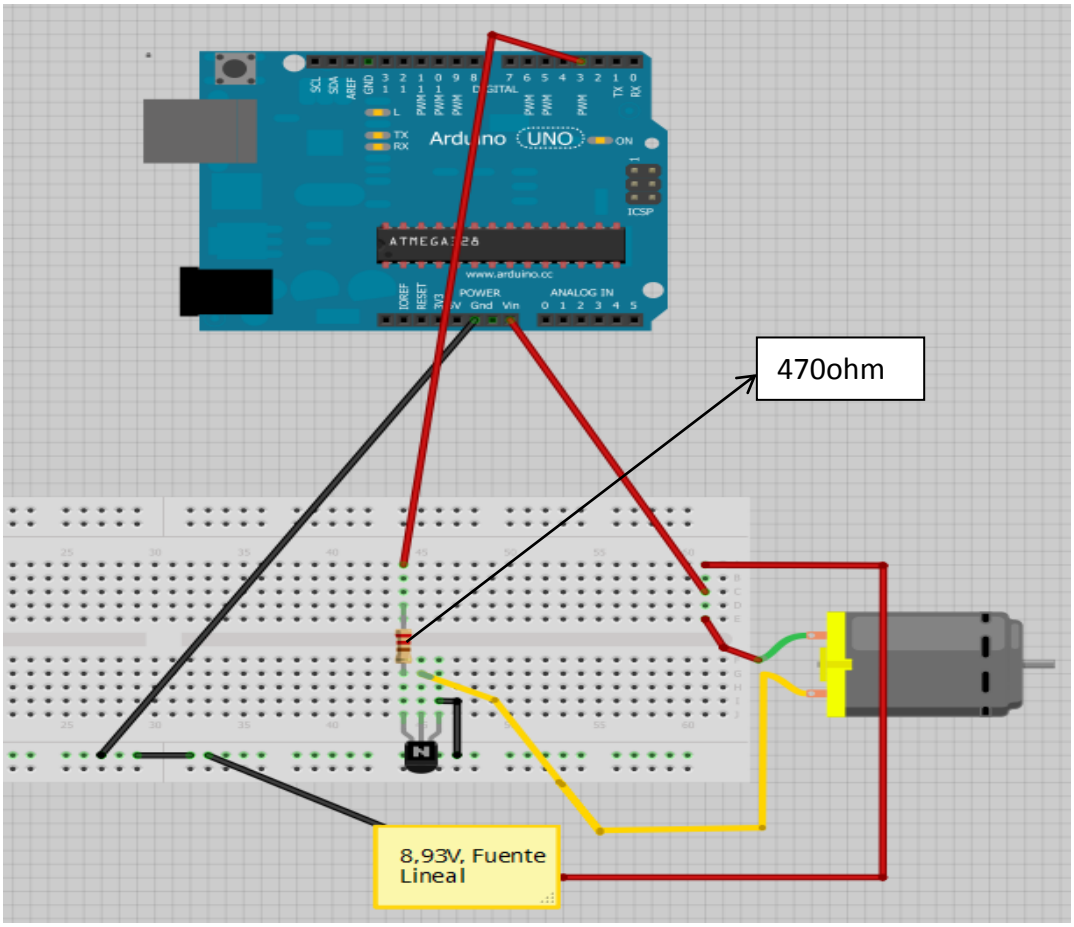


Imagen 5



Circuito 3

▪ 3.5-Ventilador

El ventilador que utilizamos fue el **KD1209PTS1 H**. El cable rojo del ventilador lo conectamos directamente a 9V CC; y el negro al colector (C) del transistor NPN.

Para poder controlar la velocidad de giro del ventilador o “cooler”, utilizamos en Arduino un procedimiento para generar una señal de modulación por ancho de pulso, conocido como **PWM (Pulse-width modulation).** Existe otro parámetro asociado 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.

Por ejemplo, si una señal tiene un periodo de 10 ms y su ancho de pulso es de 2ms, dicha señal tiene un ciclo de trabajo (duty cycle) de 20% (20% on y 80% off). En nuestro caso, como la señal de salida PWM en los pines 3,9,10,y11 de Arduino, es una señal de frecuencia **490 Hz** aproximadamente (excepto en los pines 5 y 6 que es una frecuencia de 980Hz aprox.), entonces el período T es de → T=1/f → T=1/490Hz → **2ms.** Para recordar,

la frecuencia es la cantidad de ciclos (estado on/off) por segundo; y el período mide el tiempo que se tarde en hacer un ciclo completo.

Modificando el ancho de pulso de la señal logramos variar la velocidad de giro del ventilador, de acuerdo al ciclo de trabajo que deseemos. Si utilizamos un ciclo de trabajo muy pequeño, el motor no va a alcanzar a vencer a la fuerza de inercia propia de este y no girará. Más abajo determinamos el valor mínimo en el cual el ventilador puede girar y vencer esta fuerza.

En nuestro caso queremos 3 tipos de velocidades distintas, que según la temperatura y el led que se prenda, modifique la velocidad del ventilador. Utilizamos un transistor NPN (explicado anteriormente) que actúa en modo corte o saturación. Nosotros necesitamos que esté en saturación.

Para generar la señal PWM con Arduino, hay que tener en cuenta que no todos los pines de Arduino pueden generar PWM solo los que lo indique la placa (en nuestro caso usamos el **pin 3**). Hay 2 formas de generar señales PWM, una forma es utilizando la función analogWrite(pinPWM,valor), y la otra forma es utilizando la capacidad del microprocesador. La señal de salida obtenida de un microprocesador es una señal digital de 0 Voltios (LOW) y de 5 voltios (HIGH), y con sólo realizar modificaciones en los intervalos de tiempo que el pin seleccionado tenga valor HIGH o LOW, a través de la función digitalWrite (), generamos la señal PWM.

En nuestro caso utilizaremos la primer forma, para usar el PWM simplemente usaremos la función **analogWrite(pinPWM,valor)** donde “valor” es un numero entre 0 y 255. De esta manera con distintos valores, variamos el DC; el cual si valor=0 el ventilador está siempre apagado y si valor=255 está siempre encendido.

Estos valores (0-255) están dados, debido a que **el generador PWM de Arduino (temporizador) tiene una resolución de 8 bits** por lo que tiene 256 valores distintos de codificación de señal (ya que 2^n=256, donde n es la cantidad de bits, 8 para este caso). Esto limita nuestra precisión para codificar la señal. De esta manera, de 0-255 es la cantidad de valores totales que el temporizador puede llegar a leer.

Definimos 3 Duty cycle que son:

DC=35% -> El ventilador gira a la velocidad mínima.
DC=50% -> El ventilador gira a la velocidad media.
DC=100% -> El ventilador gira a la velocidad máxima.

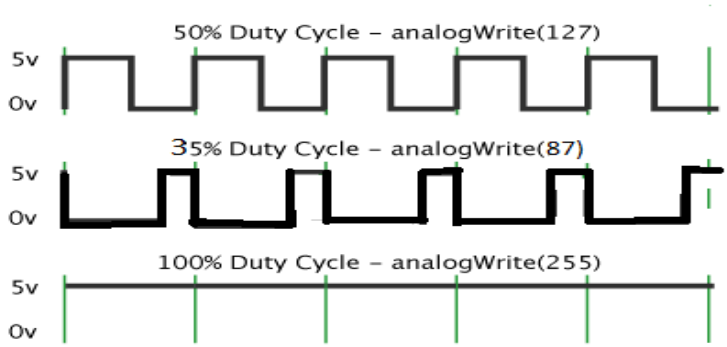


Imagen 5.1, muestra de cada DC.

Estos Duty cycle, los sacamos de una tabla normalizada (***Imagen 5.2***); a excepción del DC=35% que abajo está explicado más abajo porque usamos un value de 87.

value	Ciclo de trabajo
0	0%
63	25
127	50%
190	75%
255	100%

Imagen 5.2

En principio, teníamos en mente poner un Duty Cycle del 25% como velocidad mínima del ventilador pero nos dimos cuenta que no se podía, debido a que el motor con esta duración de ciclo no giraba y esto es debido a la fuerza de inercia del motor, que necesita un mínimo de ciclo de trabajo para poder funcionar. De forma que cambiando el valor del parámetro value en la función analogWrite(pinPWM,value), podemos obtener distintos ciclos de trabajo: este valor (Value) lo fuimos calculando probando en arduino y viendo la variación de la velocidad del motor, el value que utilizamos fue 87, ya que a valores menores no giraba el ventilador, ese valor lo pusimos cómo condición limite, y lo consideramos para un ciclo del trabajo de aproximadamente 35% al 40%.

3.6-Leds

Los diodos LED (Light Emiting Diode) son dispositivos semiconductores que emiten un haz de luz cuando se les aplica una tensión.

¿Para qué los necesitamos?: Para que prenda el led rojo si la temperatura es alta, amarillo si es media y verde si es baja.

Necesitamos un circuito independiente para cada led, al cual le deben llegar los 5V proporcionados por Arduino, y debe tener aparte del led en cuestión, una resistencia (ya que sin esta resistencia circulará una corriente desproporcional que quemará a los leds).

Los leds, soportan hasta una intensidad de corriente de 20mA (Si supera este valor se rompen y dejan de funcionar); y brillan intensamente cuando circula 15mA aproximadamente. Dicho esto, calculo las R que necesito para que los leds no se quemen y puedan brillar intensamente, dando un intervalo entre la RMAX que necesito para que circule la IMIN (15mA) y la RMIN que necesito para que circule la IMAX (19mA elegimos):

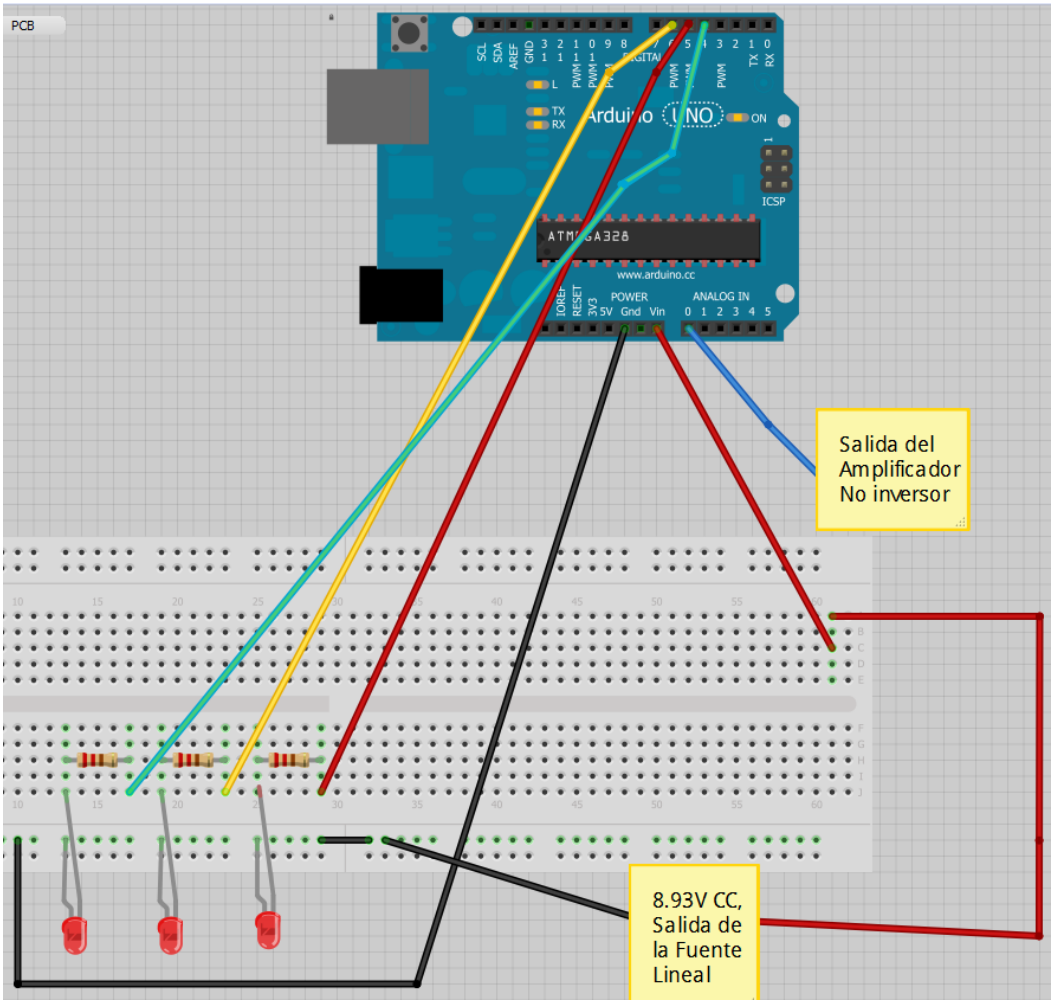
Como lo alimento con arduino, entran 5V, entonces por ley de ohm:

V=I.R → 5V=IMIN.RMAX→ 5V/15ma=333Ohm

V=I.R → 5V=IMAX.RMIN→5V/19ma=260Ohm

Entonces, necesito una R de valor entre 333ohm y 260ohm; elegimos una de **320ohm** (ya que hay resistencias de este valor). De esta forma la corriente que circulará por cada led será de→ 5V=I.320ohm → I=0,0156A= aprox. **15ma** (corriente que circulará). Como a 15mA los leds brillan intensamente, es correcto.

Luego realizamos las conexiones como lo indica la imagen del ***Circuito 4;*** y teniendo en cuenta que la pata larga del led es el positivo (ánodo), y la pata corta el negativo (cátodo). De esta manera, nuestro circuito quedó como se ve en las ***Imágenes 5.3, 5.4, 5.5 y 5.6.***



Circuito 4

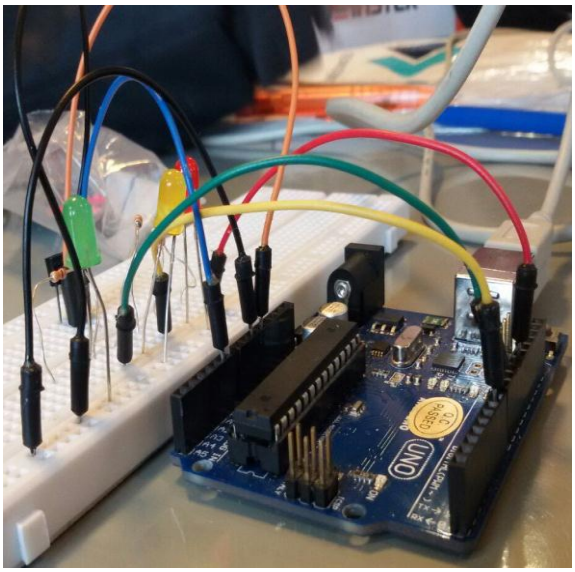


Imagen 5.3.

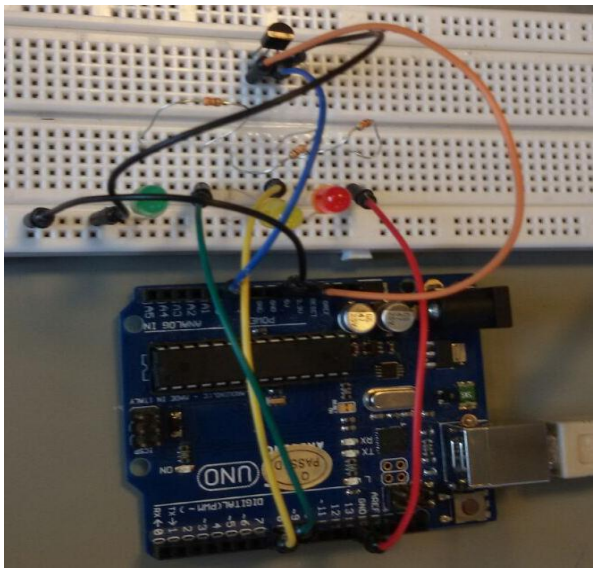


Imagen 5.4.

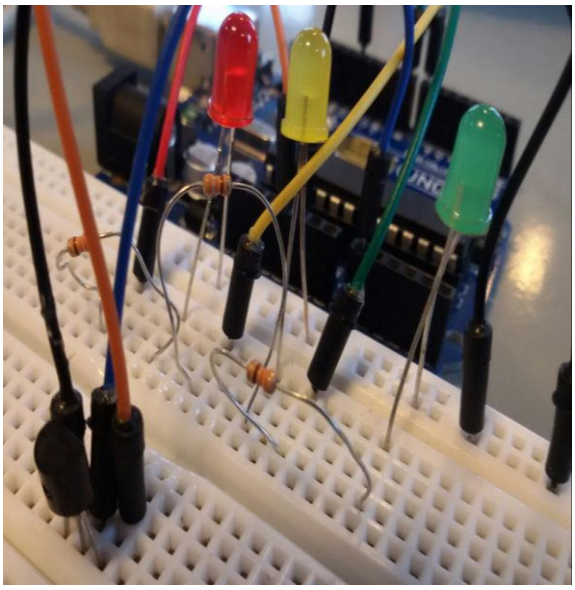


Imagen 5.6.

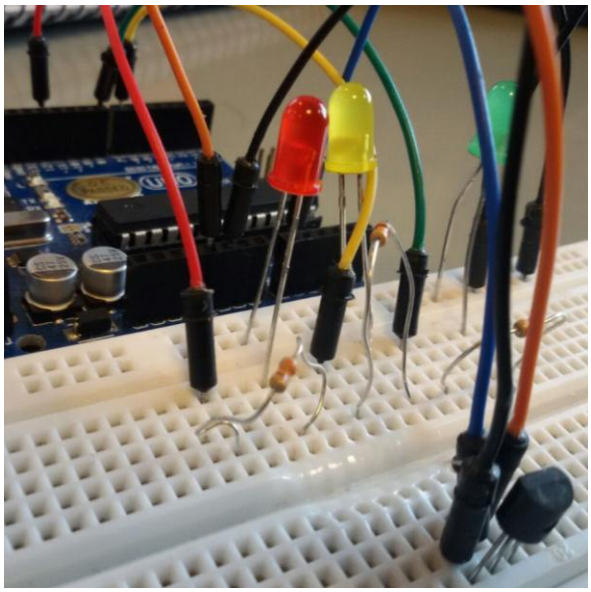


Imagen 5.5.

▪ 3.7-LCD

¿Para qué lo necesitamos?: Para poder visualizar la temperatura indicada por el sensor. Usamos el modelo gdm1602k de 16x2 (16 caracteres x 2 líneas; ó 16 columnas x 2 filas) ya que la facultad contaba con este (***Imágenes 5.6 y 5.7)***. Este modelo cuenta con 16 pines (indicados en rojo en la ***Imagen 5.6***).

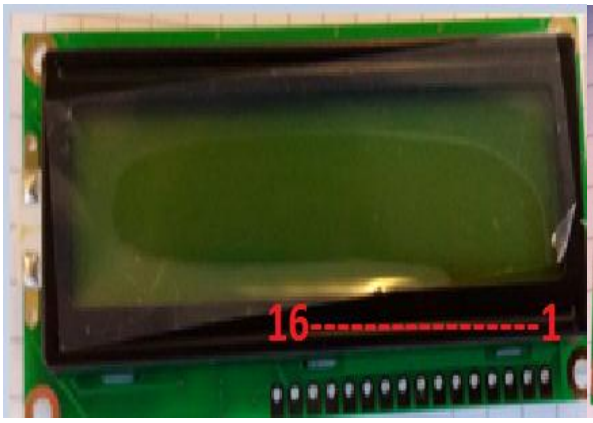


Imagen 5.6

Imagen 5.7

Para poder mostrar la temperatura tuvimos que programar en Arduino, incluyendo la librería correspondiente. Tuvimos en cuenta un rango de temperatura considerable para nuestras mediciones que fue de 0v a 50ºC. De acuerdo a ese rango programamos en arduino teniendo en cuenta que para que se enciendan los leds tiene que pasar:

T<18ºC -> Se enciende el led verde -> El ventilador gira a la mínima velocidad (ciclo de trabajo 35%)
18<=T<=22 -> Se enciende el led amarillo -> El ventilador gira a la velocidad media (ciclo de trabajo 50%)
T>22 -> Se enciende el led rojo -> El ventilador gira a la máxima velocidad (ciclo de trabajo 100%).

La conexión del LCD con el Arduino no es complicada, pero requiere de mucho cuidado. Primero conectamos el pin 16 y el pin 1 de tierra del LCD a GND y el pin 15 y 2 de alimentación del LCD lo conectamos a 5V en el protoboard.

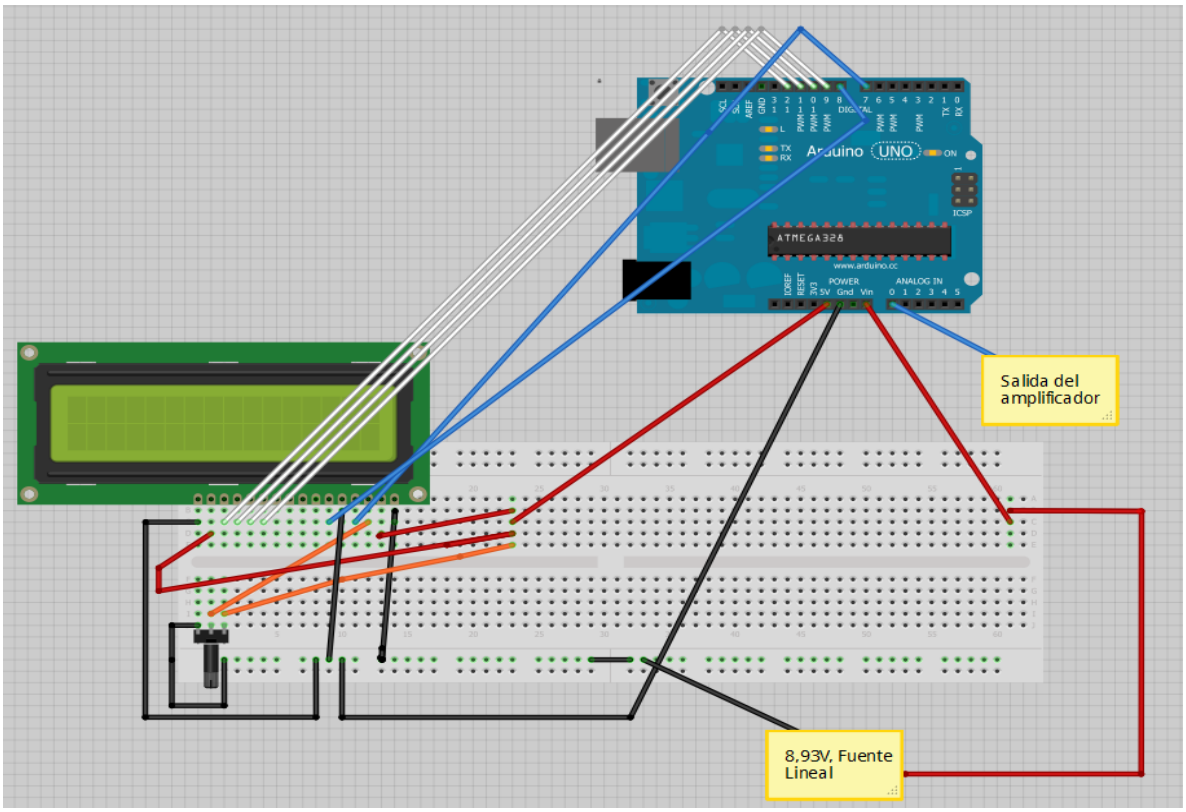
Ahora conectamos el potenciómetro de ajuste para la iluminación del LCD. Para ellos conectamos un extremo del potenciómetro a 5V y otro a GND; y en el centro lo conectamos al pin 3 del LCD.

El LCD necesita además 3 pines de control (RS, RW y EN; cables azules en nuestro circuito) y 4 pines de datos (D4, D5, D6 Y D7; cables blancos en nuestro circuito). De esta manera conectamos el RS (pin 4 del LCD) al pin 7 de arduino, el RW (pin 5 LCD) a GND, el EN (pin 6 LCD) al pin 8 de arduino, el D4 (pin 11 LCD) al pin 9 de arduino, el D5 (pin 12 LCD) al pin 10 de arduino, el D6 (pin 13 LCD) al pin 11 de arduino, y por último el D7 (pin 14 LCD) al pin 12 de arduino.

Ahora ya nos metemos en el Software, el cual está explicado en el código de Arduino (**sección 5**) con comentarios en cada línea. Brevemente, incluimos la librería para poder usar el LCD (**#include <LiquidCrystal.h>**), luego declaramos los pines de control y de datos (**LiquidCrystal lcd(7, 8, 9, 10, 11, 12);**). Luego en **setup()** fijamos las columnas y las filas (**lcd.begin(16, 2)**;), y en **loop()** imprimimos la temperatura seteando el cursor en la posición que queríamos.

En la conexión del LCD con el Arduino, utilizamos una **conexión en paralelo**. De esta manera por los pines de control y de datos nombrados anteriormente se transmiten la información (los 1s o 0s) en grupos de bits en simultáneo.

De esta manera obtenemos el ***Circuito 5;*** el cual funciona correctamente ya que nos detalla la temperatura medida (***Imagen 5.8***). Además, vemos como varía la temperatura por monitor serial mediante el puerto serie (el cable USB que conecta a la PC con arduino) en la ***Imagen 5.9.*** Estas mediciones son correctas, ya que por ejemplo para una temperatura de 19.9°C (primer ejemplo), indica que a la salida del sensor LM35 hay 199mV (ya que son 10mV por cada 1°C); dado que a la salida del amplificador son (Vi. ganancia=Vo → 199mV.9,7=Vo → Vo=1,93V), entonces es correcto.



Circuito 5

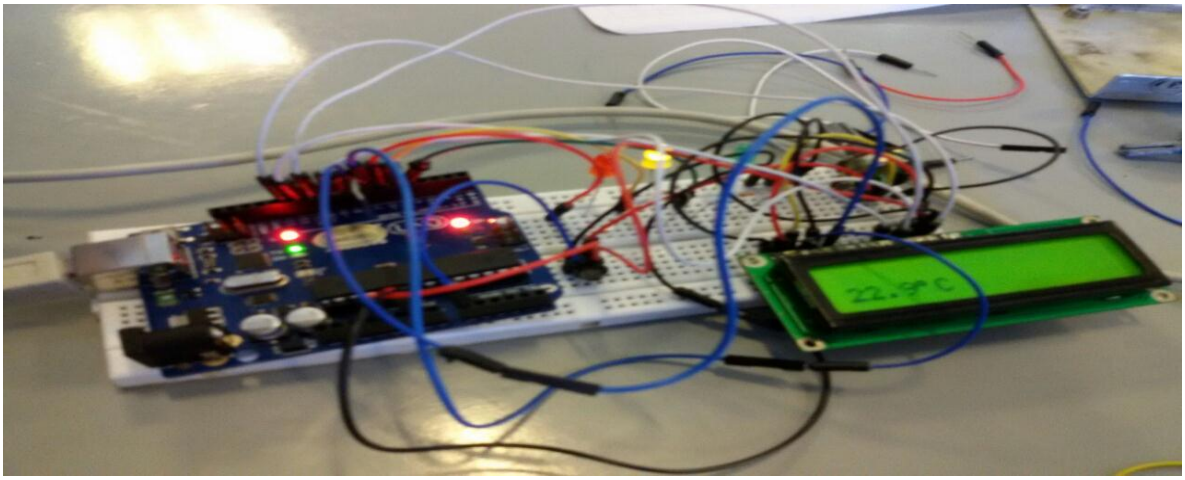


Imagen 5.8

```
Temperatura:19.90
Grados Celsius.Voltaje:1.93
Temperatura:22.32
Grados Celsius.Voltaje:2.17
Temperatura:22.47
Grados Celsius.Voltaje:2.18
Temperatura:22.57
Grados Celsius.Voltaje:2.19
Temperatura:22.57
```

Imagen 5.9.

- **4-Medición del riple en vacío y con carga.**

En un principio realizamos **2 mediciones** conectando la fuente lineal (3.1) al ventilador directamente (sólo al ventilador, no a todo el circuito; de esta manera RL es el ventilador). La conexión que realizamos (ver ***Imagen 6***) fue conectar el cable naranja de salida de la fuente con el rojo del ventilador ; y el cable negro de la salida de la fuente con el negro del ventilador → en esta última unión colocamos la tierra del osciloscopio (***Imagen 6.1***), círculo azul de la ***Imagen 6***.

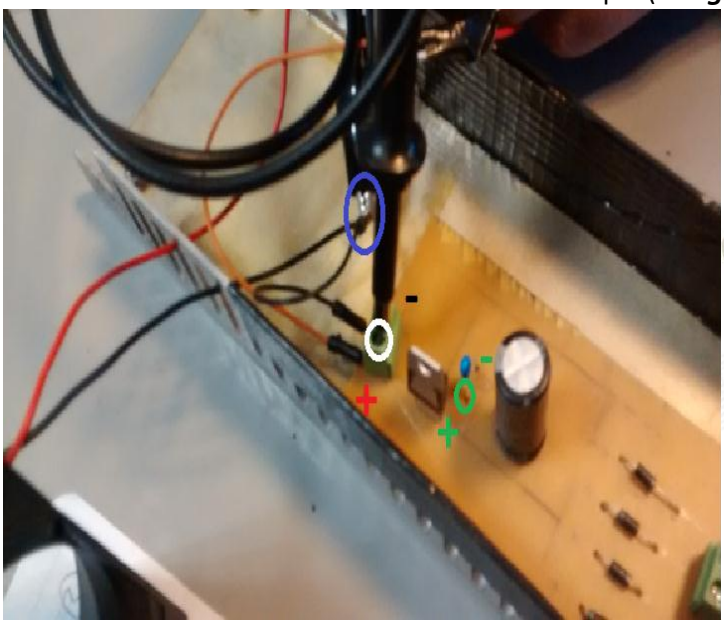


Imagen 6

Imagen 6.1

La **medición 1** la realizamos midiendo ANTES del regulador de V, en el capacitor de 0.1uF (***Imagen 6.2***); así pusimos la punta del osciloscopio en el círculo verde (+ del capacitor de 0.1uF) y tierra en el circulo azul explicado anteriormente.

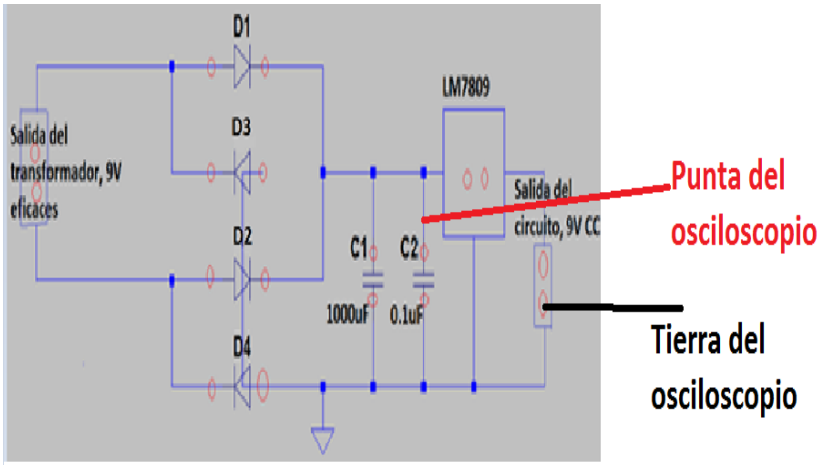


Imagen 6.2

Observamos en el osciloscopio el riple al “sacarle” la parte continua a la señal y dejándole la componente de alterna (el riple) en la ***Imagen 6.3***. Este riple es **bajo.**

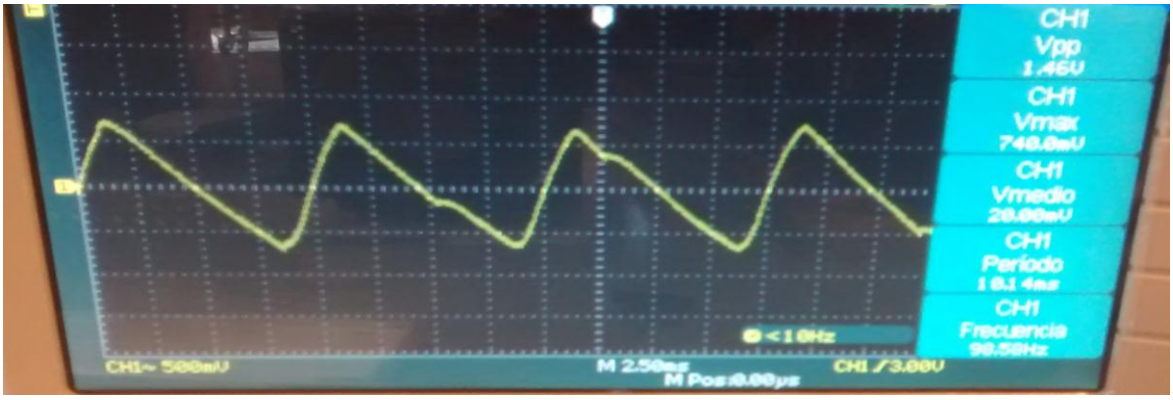


Imagen 6.3.

La **medición 2** la realizamos midiendo DESPUES del regulador de V, osea a la salida del circuito (***Imagen 6.4***); así pusimos la punta del osciloscopio en el circulo blanco (la salida + del circuito) y tierra igual que en la medición 1.

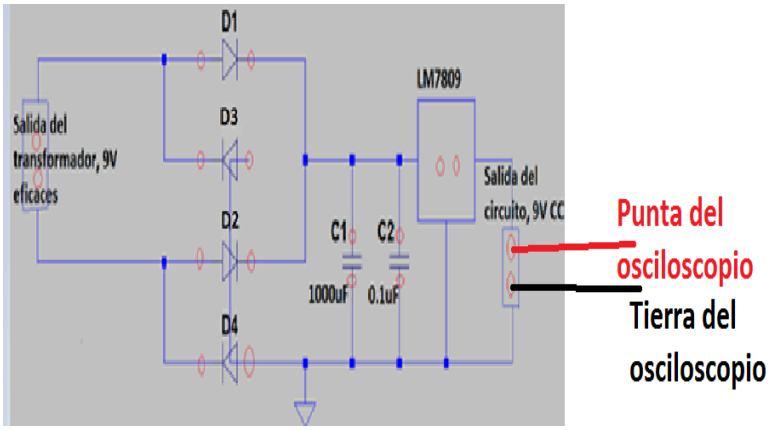


Imagen 6.4.

Observamos que el regulador **bajó aún más el riple (*Imagen 6.5*);** y que se produjeron unos **picos**; la causa de estos es el regulador que los produce en el proceso de mantener fijo el voltaje.

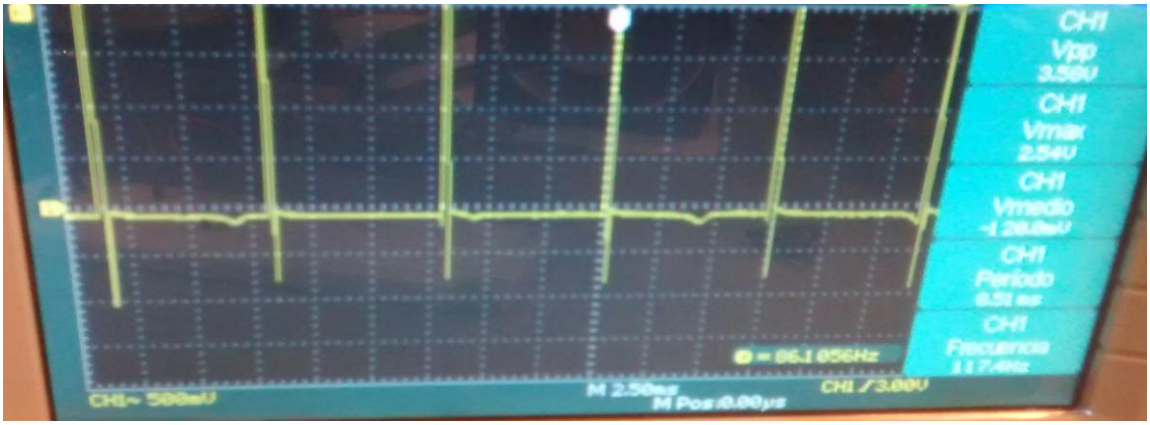


Imagen 6.5

Luego, realizamos **2 mediciones más**, en la 1ra (***Imagen 6.6***) medimos el riple en vacío, y en la 2da (***Imagen 6.7***) medimos el riple conectando la salida de la fuente lineal a todo el circuito funcionando en la condición "crítica" (donde el ventilador exigía más corriente), cuando su DC=100% (prendiéndose el led rojo).

Como sabemos, al rectificar una señal a corriente continua, se produce ripple o rizo. Este mismo puede reducirse notablemente mediante un filtro capacitivo, este proceso es llamado "filtrar", y debe entenderse como la reducción a un valor mucho más pequeño de la componente alterna tras la rectificación (por nuestros 4 diodos). Elegimos un capacitor de 1000uF ya que necesitábamos un capacitor grande para reducir el ripple (la pequeña variación de corriente alterna que queda tras rectificarse el circuito). En los instantes en que los diodos no conducen, (el potencial del cátodo es mayor al del ánodo, polarización en inversa) se comportan cómo un circuito abierto. Mientras dure este estado el capacitor se descargará sobre la "resistencia" (en nuestro caso esta "resistencia" es el Arduino y el ventilador). Pero cuando los diodos conducen (potencial de ánodo mayor al del cátodo, polarización directa) se volverá a cargar el capacitor. Cómo en este instante no tenemos resistencia ya que estamos midiendo a la salida del circuito rectificador, no hay ripple porque el capacitor no se descarga porque no tiene resistencia en donde descargarse. Cómo se puede apreciar en la imagen a continuación, casi no hay variación de ripple, es decir es muy pequeño. Es lo que se conoce como condición de ripple o rizado en vacío **(medición 1).**

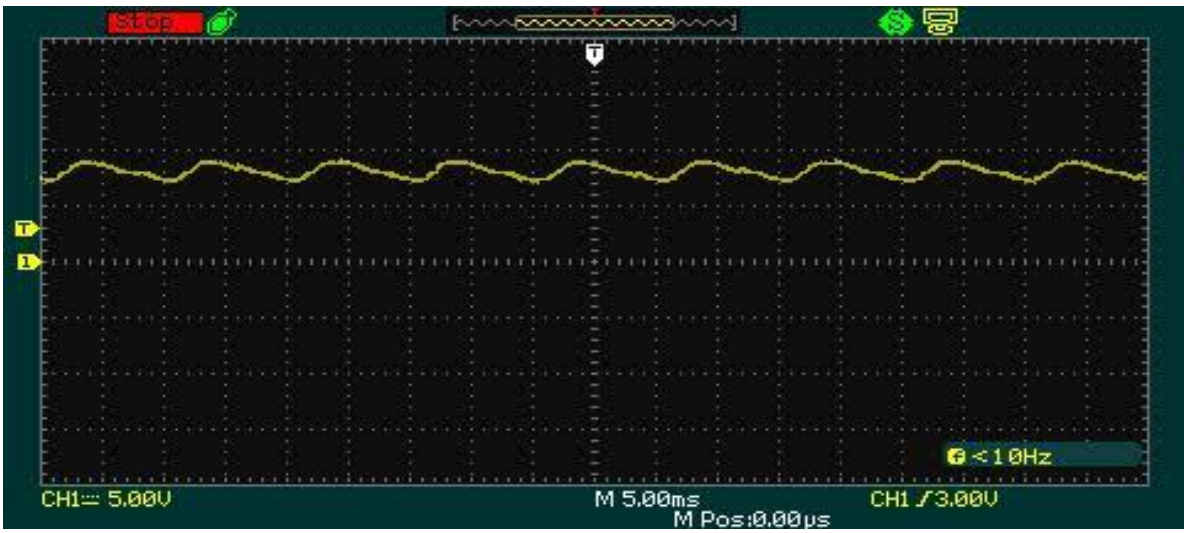


Imagen 6.6

Para la **medición 2**, conectamos la fuente lineal (circuito 3.1), para alimentar al ventilador y al arduino. Al conectarlo, tenemos un consumo, ese consumo va a ser que se produzca ripple, ya que el capacitor ahora si se va a descargar y cargar sucesivamente. El voltaje de rizado es el voltaje de corriente alterna que está presente dentro de la componente de CC y es causada por la carga y descarga del capacitor de filtrado (nuestro capacitor de 1000uF). Este ripple puede ser mayor o menor dependiendo del consumo que tengamos. Cómo se puede ver en la imagen el ripple tiene un Vpp=2.18 V. La escala vertical es de 500mv.

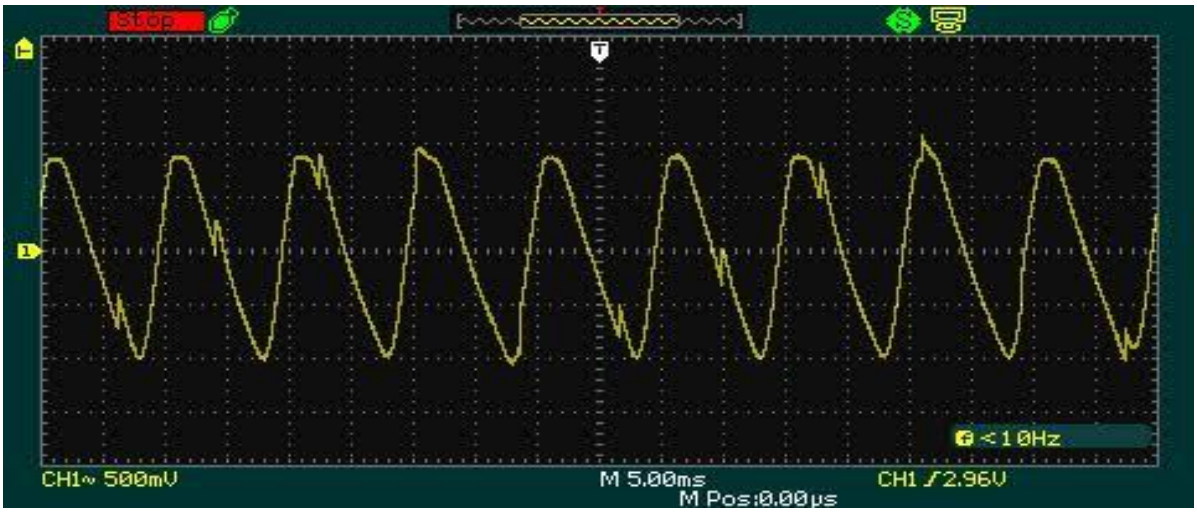


Imagen 6.7

Una forma de bajar el ripple, es poniendo un capacitor más grande, en nuestro caso es de 1000uF, cambiando por uno más grande lograríamos bajar aún más el ripple (esto está explicado en “mejoras”).

5-Programación y Software

¿A qué llamamos Software?: Arduino brinda un entorno de desarrollo(IDE), es el software que te permite programar y cargar el código en el microcontrolador que se encuentra en la placa de Arduino. Se dispone de versiones para Windows y para MAC, así como las fuentes para compilarlas en LINUX. Lo primero que tenemos que hacer para comenzar a trabajar con el entorno de desarrollo de arduino es configurar las comunicaciones entre la placa Arduino y la PC. Para ello deberemos abrir en el menú "Tools" la opción "Serial Port". En esta opción deberemos seleccionar el puerto serie al que está conectada nuestra placa. Luego mediante distintas funciones, controlamos nuestro circuito; estas funciones y el código final que utilizamos para este proyecto, está detallado en el ***Apéndice (Sección 9);*** donde al lado de cada línea comentamos para que sirven dichas funciones.

6-Problemas

Tuvimos varios problemas, el mayor de estos fue controlar la velocidad del ventilador y el conexionado que implicaba. Teníamos mal conectado el ventilador con la patita colector del transistor la cual sin querer la estábamos poniendo en corto por no estar correctamente ubicada. Por este motivo el transistor no estaba conduciendo, lo cual no estaba saturando, por lo tanto el transistor no estaba trabajando cómo debía. Al tener un problema en el hardware, también se tiene un problema en el software, ya que no se podía controlar bien el duty cycle (ciclo de trabajo, DC) porque había un problema en el circuito. Tardamos en darnos cuenta de este problema, ya que el ventilador giraba y parecía que estaba bien conectado pero en realidad no lo estaba. Cuando descubrimos esto con ayuda del profesor, pudimos manejar bien el DC, y controlar correctamente la velocidad, ya que de este modo, nos habíamos asegurado que el circuito andaba bien (hicimos las mediciones correspondientes y la comparamos con los cálculos) y de este modo pasó a ser problema del software y no del hardware.

- # 7-Mejoras.

Siempre en todo proyecto hay mejoras para hacer, en nuestro caso como posibles mejoras son:

1-Usar una fuente de continua de 5v para alimentar al display LCD, ya que la corriente que consume es proporcionada por un pin del microcontrolador, lo cual es conveniente NO manejar corrientes con los pines del micro, ya que se le exige más corriente al Arduino de la que puede dar. Si bien funciona todo correctamente, es aconsejable tomar esas precauciones, ya que nosotros teníamos muchos elementos conectados al Arduino directamente (leds, el LCD mencionado, amplificador, transistor, sensor de temperatura). No utilizamos una fuente de continua de 5v para alimentar al display LCD, ya que se hacía "molesto" tener que trasladar la fuente que es muy pesada y ocupa mucho lugar, y no había problema en alimentarla desde arduino.

2-Conseguir un rizo más bajo aún, poniendo un capacitor más grande, en nuestro caso es de 1000uF, cambiando por uno más grande (ó poniendo otro capacitor en paralelo) lograríamos bajar aún más el ripple.

Recordando, el factor de rizado o ripple es un indicador de la efectividad del filtro que se define cómo: r = (Vr / Vcd) x 100% → donde: Vr es el voltaje de rizado eficaz (RMS) y Vcd es el valor de CD (corriente continua promedio) del voltaje de salida del filtro. Cuanto menor sea el factor de rizado, mejor será el filtro. **El factor de rizado puede reducirse incrementando el valor del capacitor del filtro, como se mencionó anteriormente.**

3-Cambiar el transformador por uno de mayor V de salida, ya que estamos un poco limitados con los 9,9V eficaces del transformador que usamos.

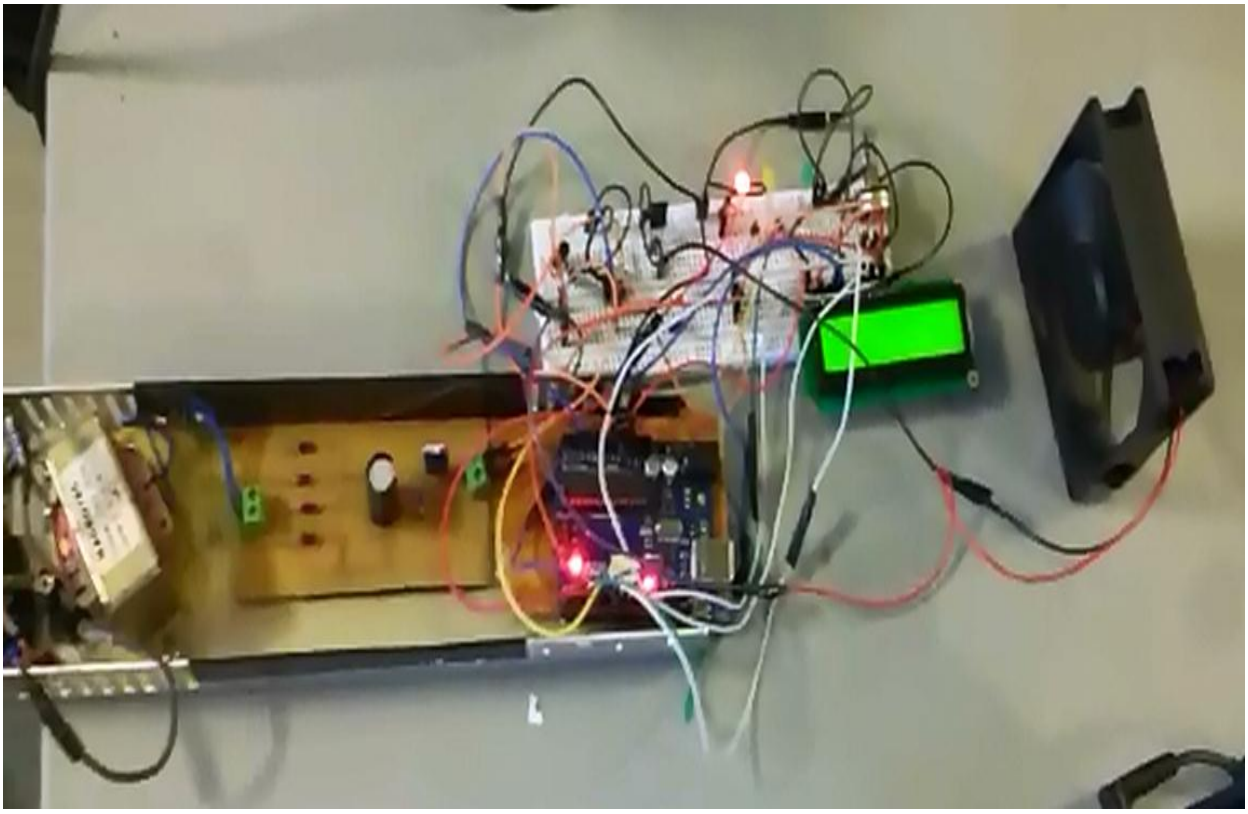
4-Colocarle un VCC negativo al amplificador no inversor utilizado, para poder medir temperaturas negativas.

5-Mejorar el Circuito impreso, ya que lo hicimos a mano pero se podría hacer con los software Fritzing o KICAD.

- **8-Conclusiones y foto final del circuito.**

Como cierre final del proyecto podemos decir que pudimos terminar en tiempo y forma con la realización del mismo, funcionando correctamente. Logramos integrar los temas vistos en clases, y de materias anteriores como son: Análisis de circuitos, y STE.

Foto final del circuito:



- **9-Apéndice.**

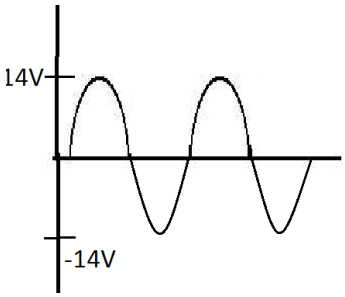
Código que utilizamos en el IDE de Arduino para nuestro proyecto:

```
#include <LiquidCrystal.h> //Librería para el LCD
LiquidCrystal lcd(7, 8, 9, 10, 11, 12); //    LCD: pines de control y de datos (RS, EN, d4, d5, d6, d7)
//Entrada de todos los pines:
const int Sensor =0;       //Entrada del sensor de temperatura LM35
int ledrojo=5;
int ledamarillo=6;
int ledverde=4;
float voltaje;             //Variable de calculo
float temperatura;         //Variable de resultado final
int PinPWM = 3;            //Para el ventilador, para poder variar el DC

void setup() {
          pinMode(ledrojo,OUTPUT);
          pinMode(ledamarillo,OUTPUT);
          pinMode(ledverde,OUTPUT);
          Serial.begin(9600); //Para ver los serialPrint por Monitor Serial, 9600 es la velocidad de muestreo.
          lcd.begin(16, 2);   //Para fijar el numero de columnas y de filas del LCD.
}
void loop ()
{         float lectura = analogRead(Sensor);
          voltaje = (5.0/1023.0) * lectura; //Convierte la lectura analogica de 0-1023V a un Voltaje de 0-5 volts.
          temperatura = (voltaje*100)/9.7 ; //Calculamos la temperatura, si salian 1,76V de la salida del amplificador, la Temperatura
          //es de 18°C (por la ganancia del amplificador que es de 9,7 aprox.).

          //Para visualizar la temperatura por el monitor serial:
          Serial.print("Temperatura:"); //Imprime "temperatura" para verlo por el monitor serial.
          Serial.println(temperatura); //Imprime la temperatura con un SALTO de linea (si fuera solo Serial.print(temperatura) no lo hace).
          Serial.print("Grados Celsius.");
          //Para visualizar el voltaje por el monitor serial:
          Serial.print("Voltaje:");
          Serial.println(voltaje);


        //Control de la temperatura:
        if (temperatura<18){
          digitalWrite (ledverde, HIGH);    //Prendo el led verde.
          digitalWrite (ledamarillo, LOW); //Apago el led amarillo.
          digitalWrite (ledrojo, LOW);      //Apago el led rojo.
          analogWrite(PinPWM,87);           //VENTILADOR A 8.6V, DC=35%.
      }
       if (temperatura>=18 && temperatura<=22){
          digitalWrite (ledverde, LOW);
          digitalWrite (ledamarillo, HIGH);
          digitalWrite (ledrojo, LOW);
          analogWrite(PinPWM,127);          //VENTILADOR A 8.6V,DC=50%.
      }
       if (temperatura>22){
          digitalWrite (ledverde, LOW);
          digitalWrite (ledamarillo, LOW);
          digitalWrite (ledrojo, HIGH);
          analogWrite(PinPWM,255);         //VENTILADOR A 8.6V,DC=100%.
     }
       delay(2000);//Para evitar errores, se esperan 2000ms (2 seg) para volver a tomar la temperatura.
          //LCD:
          lcd.setCursor(0, 0);         // Ubicacion del printeo de abajo, columna 0, fila 0.
          lcd.print("Temperatura:");   // Printeo "temperatura".
          lcd.setCursor(0, 1);         // Ubicacion del printeo de abajo.
          lcd.print(temperatura);      // Printeo la temperatura sea amarillo, verde o rojo.
          lcd.setCursor(4, 1);         //Columna 4, fila 1.
          lcd.print((char)223);        //Printeo °
          lcd.setCursor(5,1);          //Columna 5, fila 1.
          lcd.print('C');              //Printeo C
```

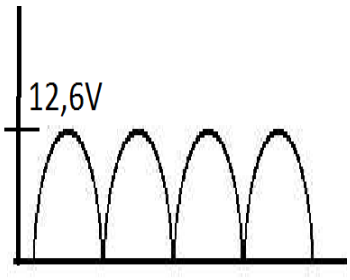
APARTADO MEDICIONES:

Componente	Volts entrantes	Volts salientes
Transformador	220V CA eficaces (311 V pico)	9,9 9V CA eficaces (14V pico)
Circuito Impreso	9,9V CA eficaces (14 V pico)	8,93V CC
Regulador de Voltaje	12,6 V CA	8,93V CC

Salida del transformador:



Salida del circuito tras rectificarse, voltaje que entra al regulador:



Al conectar el regulador de voltaje LM7809 a la salida del circuito rectificador (***Imagen 6.8***), teníamos que ver si como mínimo llegaba a 11V para regular bien (***Tabla 1***). Como en las mediciones obtuvimos un valor de 12,6V CA entonces SI lo pudimos conectar y reguló bien a aprox. 9V CC (en las mediciones obtuvimos 8,93V CC).

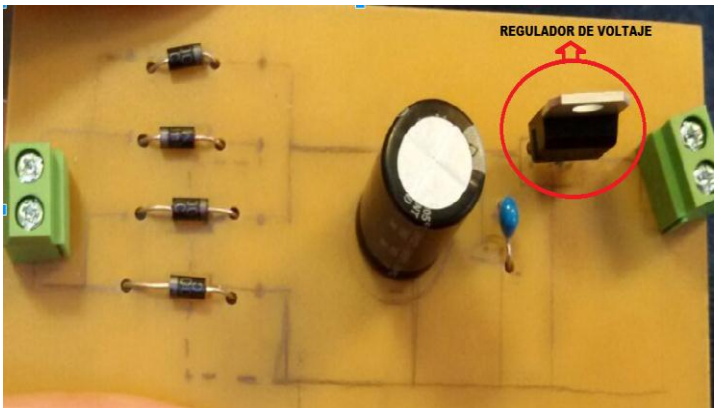


Imagen 6.8

Type	Min Input Voltage	Output Voltage
7805	7V	+5V
7806	8V	+6V
7808	10V	+8V
7809	11V	+9V
7812	15V	+12V
7815	18V	+15V
7818	22V	+18V
7824	30V	+24V

Tabla 1