# Laboratorio 4: Entradas y Salidas Analógicas

## 1. Datos de la Practica

| | | | |
|---|---|---|---|
| Carrera | INGENIERÍA ELECTRÓNICA | | |
| Semestre | | Grupo | |
| Tipo de Práctica | ☐ Laboratorio ☐ Simulación | Fecha | |
| Asignatura | Curso Arduino | | |
| Unidad Temática | | | |
| Nº Alumnos por práctica | 2 | Nº Alumnos por reporte | 2 |
| Nombre del Profesor | | | |
| Nombre(s) de Alumno(s) | 1.<br>2. | | |
| Tiempo estimado | | Vo. Bo. Profesor | |
| Comentarios | | | |

## 2. Objetivo

- Familiarizarse con el módulo ADC del Atmega 2560 para las entradas analógicas.

## 3. Componentes a Utilizar

Por cada práctica y por cada puesto de laboratorio, los materiales a utilizar son:

| Cantidad | Descripción |
|---|---|
| 1 | Computadora |
| 1 | Arduino MEGA |
| 1 | LEDs |
| 1 | Resistencia de 330 ohm |
| 2 | Resistencia de 10K |
| 1 | LCD 16x2 |
| 1 | LM35 |

## 4. Introducción

### Entradas Analógicas

El mundo que nos rodea es completamente analógico por lo tanto es necesario un dispositivo que nos convierta estas magnitudes físicas (distancia, calor, luz, etc.) a un valor eléctrico (voltaje o corriente) entendible para los dispositivos electrónicos. Estos dispositivos son llamados transductores y la gran mayoría de ellos son sensores, por ejemplo, sensores de temperatura, detectores de presión, detectores de proximidad, etc. El transductor puede conectarse directamente a uno de los pines analógicos del Arduino y este es capaz de leer valores de tensión comprendidos entre 0 y 5V, si el transductor suministra otros valores diferentes necesitaríamos poner entre la salida de él y la entrada analógica un conversor adecuado.

El **Atmega 2560** presenta un convertidor Analógico a Digital (ADC) de tipo Aproximaciones Sucesivas de 10 bits, que está conectado internamente a un multiplexor de 16 canales los cuales se identifican por las letras ADC. Cuatro pares de entradas diferenciales (**ADC1 & ADC0, ADC & ADC2, ADC9 & ADC8 y ADC11 & ADC10**) están equipadas con una etapa de ganancia programable, con pasos de amplificación de 0dB(1X), 20dB(10X) o 46dB(200X) antes de conectarse al convertidor. Los 16 canales se dividen en dos secciones de 8, donde 7 entradas comparten una terminal negativa y otros dos (ADC1/ADC9) pueden ser seleccionados como terminal positiva.

El ADC dentro del microcontrolador es capaz de tener valores numéricos entre 0 y 1023, es decir toma 1024 muestras de la señal analógica que se le ingrese, teniendo la posibilidad de Multiplexar varias entradas y conectar varias señales analógicas a la vez que provengan de distintos transductores (sensores).

## Salidas Analógicas

Al igual que sucedía con las entradas analógicas, es necesario poder gobernar un determinado dispositivo con una señal analógica, por ejemplo, la intensidad luminosa de un foco, controlar el caudal de un líquido, la velocidad de un motor, etc., por lo que ahora se utiliza un convertidor de digital a analógico que se encarga de asignar valores de tensión correspondiente entre el rango de 0 y 5 voltios. En nuestro caso el Arduino no lo posee, por lo que se tendrá que hacer uso de otra técnica diferente llamada Modulación de Ancho de Pulso (PWM), en la cual una salida digital funcionará como una salida analógica modificando el ancho del pulso de encendido (ciclo de trabajo) de una señal cuadrada para obtener un porcentaje de voltaje analógico.

Por ejemplo, según la **Fig. 1**, si hacemos que el 0% del tiempo la señal esté en el nivel bajo y el 100% en el nivel alto, obtendremos una señal de 5V. Si hacemos que el 25% del tiempo esté en el nivel bajo y el 75% en el nivel alto, obtendremos una señal de 3,75V. Con la señal al 50% de nivel alto y bajo, tendremos una señal de 2,5V. Con la señal al 75% de nivel bajo y 25% de nivel alto, tendremos una señal de 1,25V. Y si la señal está el 100% del tiempo a nivel alto, la señal de salida será de 0V.

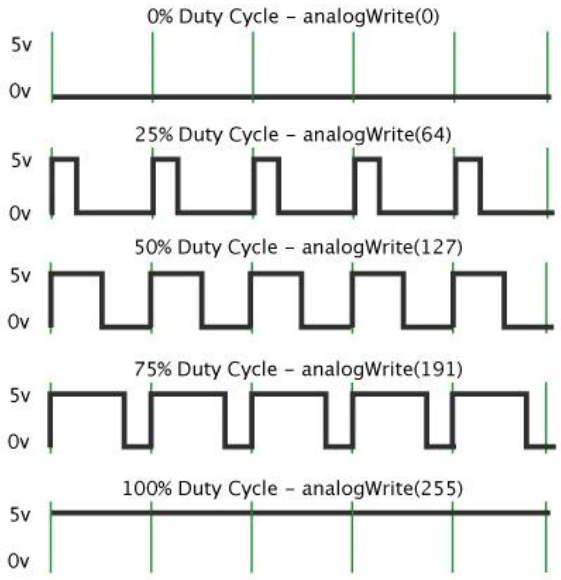


Fig. 1 Ejemplos de Porcentajes de Duty Cycle

## Timer en PWM por Hardware

Las funciones PWM por hardware emplean los Timer para generar la onda de salida. Cada Timer da servicio a 2 o 3 salidas PWM. Para ello dispone de un registro de comparación por cada salida. Cuando el tiempo alcanza el valor del registro de comparación, la salida invierte su valor. Cada salida conectada a un mismo temporizador comparte la misma frecuencia, aunque pueden tener distintos Duty Cycles, dependiendo del valor de su registro de comparación.

| Numero de Timer | Arduino UNO, Mini y Nano | Arduino MEGA |
|---|---|---|
| **Timer 0 (8 Bits)** | 5 y 6 | 4 y 13 |
| **Timer 1 (16 Bits)** | 9 y 10 | 11 y 12 |
| **Timer 2 (8 Bits)** | 3 y 11 | 9 y 10 |
| **Timer 3 (16 Bits)** | | 2, 3 y 5 |
| **Timer 4 (16 Bits)** | | 6, 7 y 8 |
| **Timer 5 (16 Bits)** | | 44, 45 y 46 |

## Frecuencia del PWM

La frecuencia de cada PWM depende de las características del temporizador al que está conectado, y de un registro de pre escalado, que divide el tiempo por un número entero. La frecuencia de los PWM se puede modificar cambiando el pre escalado de los Timer correspondientes.

Arduino Mega añade tres temporizadores más: Timer 3, 4 y 5, con una frecuencia de 31250Hz, y pre escalados de 1, 8, 64, 256, and 1024. Por tanto, la frecuencia estándar para las salidas PWM es de 490Hz para todos los pines en Arduino, excepto para el 5 y 6 (Uno, Mini y Nano) y 4 y 13 (MEGA), cuya frecuencia es de 980Hz.

El uso de los Timer no es exclusivo de las salidas PWM, sino que es compartido con otras funciones. Emplear funciones que requieren el uso de estos Timer supondrá que no podremos emplear de forma simultánea alguno de los pines PWM. Por ejemplo, la librería servo hace un uso intensivo de temporizadores por lo que, mientras la estemos usando, no podremos usar algunas de las salidas PWM.

En el caso de Arduino Uno, Mini y Nano, la librería servo usa el Timer 1, por lo que no podremos usar los pines 9 y 10 mientras usemos un servo. En el caso de Arduino Mega, dependerá de la cantidad de servos que empleemos.
Si usamos menos de 12 servos, usa el Timer 5, por lo que perderemos los pines 44, 45 y 46.
Para 24 servos, usa los Timer 1 y 5, por lo que perderemos los pines 11, 12, 44, 45 y 45.
Para 36 servos, usa los Timer 1,3 y 5, perdiendo los pines 2, 3, 5, 11, 12, 44, 45, 46.
Para 48 servos, usa los Timer 1, 3, 4 y 5, perdiendo todos los pines PWM.

## Programación del Arduino

Las dos instrucciones para leer o escribir señales analógicas son: analogRead() y analogWrite()

## 5. Prácticas de Laboratorio

## Practica #1 Lectura de un Pin Analógico

## Código Arduino

En esta primera práctica se hará la lectura de un pin analógico y dicha lectura servirá para definir el tiempo de apagado y encendido de un led.

```
int sensorPin = A0;    // Entrada para el Potenciometro
int ledPin = 2;      // Pin para el LED
int sensorValue = 0;  // variable para almacenar el valor medido

void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);
  delay(sensorValue);
  digitalWrite(ledPin, LOW);
  delay(sensorValue);}
```

## Simulación en Proteus

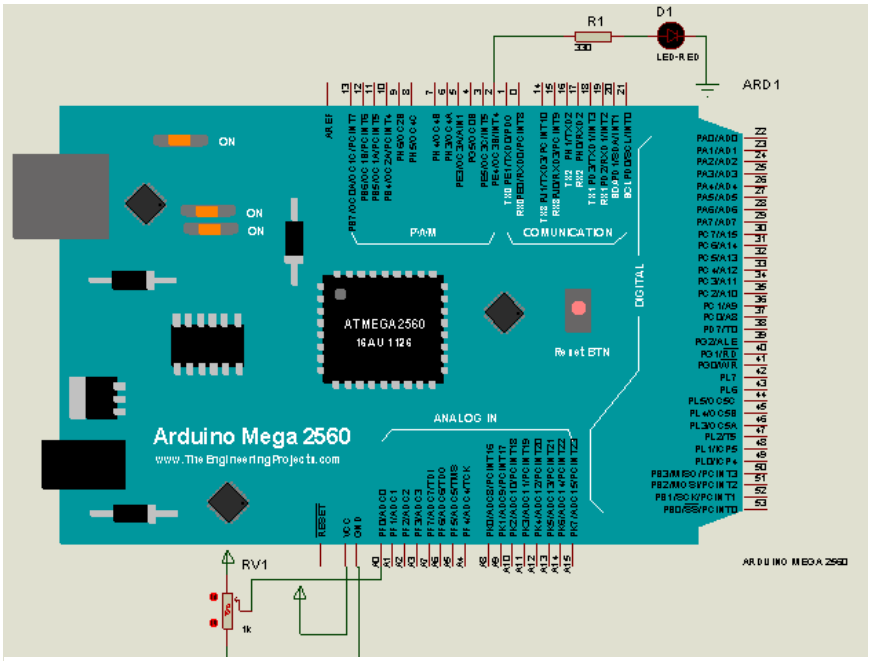


Fig. 2 Simulación de Lectura de un Pin Analógico

## Montaje del Circuito

El circuito previamente simulado en Proteus, tendrá que ser montado a como se muestra en la Figura, teniendo en cuenta dos cosas: El uso adecuado de la tabla de nodos y la identificación de Ánodo (pin más largo) y Cátodo del led. El resultado será el mismo que la simulación.

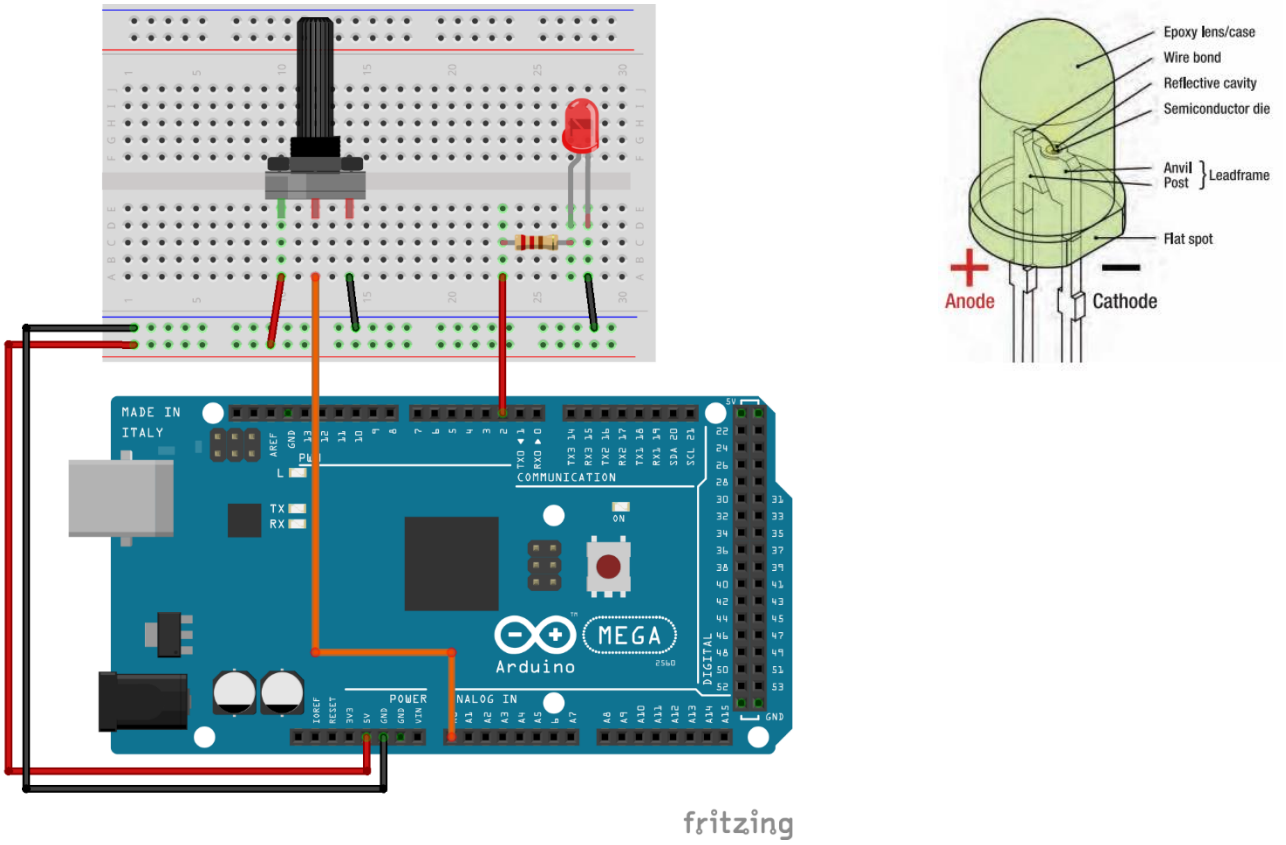


Fig. 3 Diagrama de Montaje de la Lectura Analógica.

## Actividad

Realice cambios en el potenciómetro y vea cómo afecta el parpadeo del led en el circuito.

## Practica #2 Salidas PWM para comportarse como Analógicas

## Código Arduino

La Modulación PWM será implementada para generar salidas con un comportamiento casi analógico, a través de la modificación del ciclo de trabajo de una señal periódica, en este caso una señal cuadrada. El ciclo de trabajo de una señal es el ancho relativo de su parte positiva en relación con el periodo, es decir $duty\ cycle = t_{ON}/T$ . La siguiente práctica se hará la modificación del duty cycle mediante un potenciómetro y se visualizará en el osciloscopio.

```
int sensorPin = A0;    // Entrada para el Potenciometro
int ledPin = 2;      // Pin para el LED
int sensorValue = 0;  // variable para almacenar el valor medido

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Lectura de la Entrada Analogica:
  sensorValue = analogRead(sensorPin)/4;
  // Encendido del led
  analogWrite(ledPin, sensorValue);
}
```

## Simulación en Proteus

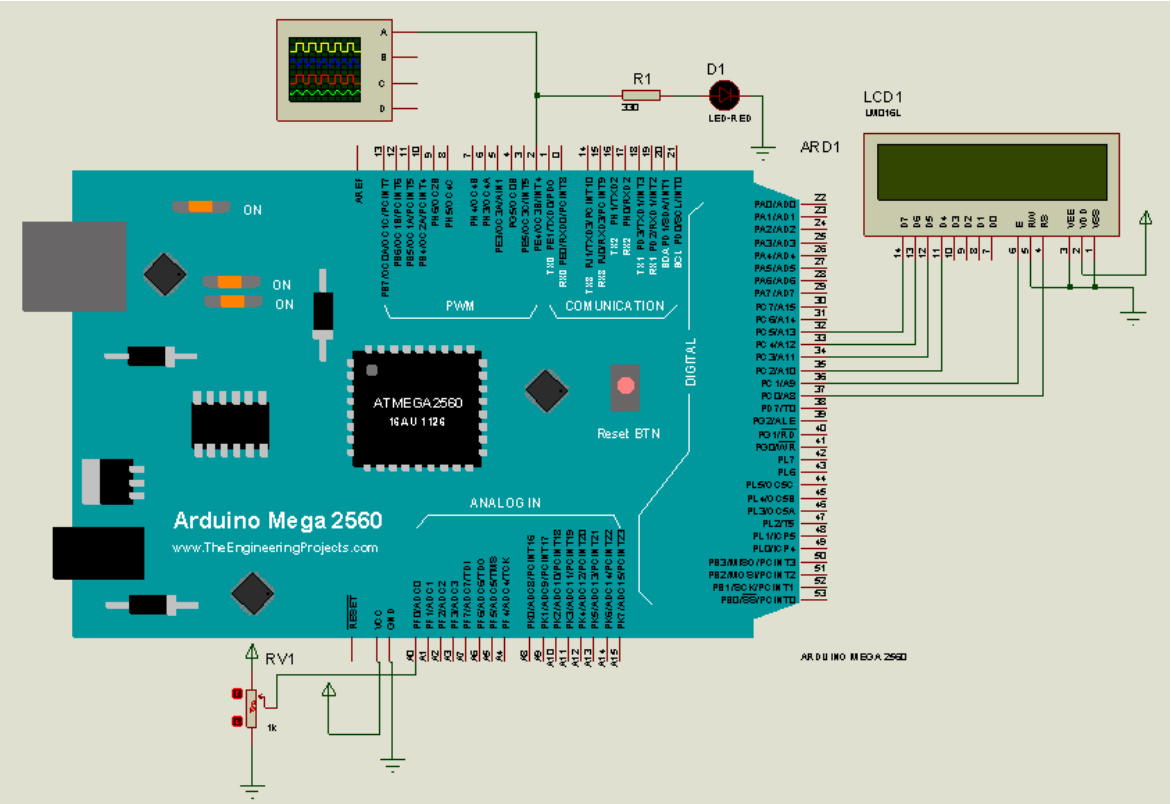


Fig. 4 Simulación de Salidas Analógicas usando PWM.

## Montaje del Circuito

El montaje para esta práctica es similar a la práctica anterior teniendo en cuenta que a la salida del led se conectará a la sonda hacia el osciloscopio.

## Practica #3 Termómetro Digital con el Sensor LM35

## Código Arduino

En esta última práctica se hará uso de un sensor de temperatura LM35 conectado a una entrada analógica del Microcontrolador, la cual será leída e interpretada para visualizarse la temperatura actual en una pantalla LCD 16x2. Tome en consideración que la instrucción *lcd.print* no imprime valores numéricos, es decir se hará la conversión del valor numérico a cadena con la instrucción *String.*

El LM35 es un sensor de temperatura con una precisión calibrada de 1 °C. Su rango de medición abarca desde -55 °C hasta 150 °C. La salida es lineal y cada grado Celsius equivale a 10 mV, por lo tanto: 150 °C = 1500 mV y -55 °C = -550 mV. Puede operar de 4v a 30v y presenta las siguientes características:

- Está calibrado directamente en grados Celsius.
- La tensión de salida es proporcional a la temperatura.
- Tiene una precisión garantizada de 0.5 °C a 25 °C.
- Baja impedancia de salida y baja corriente de alimentación (60 µA).
- Bajo coste

```
#include <LiquidCrystal.h>
const int rs = 37, en = 39, d4 = 35, d5 = 33, d6 = 31, d7 = 29;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

float tempC; // Variable para almacenar el valor obtenido del sensor
(0 a 1023)
int pinLM35 = 0; // Variable del pin de entrada del sensor (A0)

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  // Con analogRead leemos el sensor, recuerda que es un valor de 0 a
1023
  tempC = analogRead(pinLM35);

  // Calculamos la temperatura con la fórmula
  tempC = (5.0 * tempC * 100.0) / 1024.0;
  lcd.clear();
  lcd.print("Temp= "+String(tempC)+"C"); // Imprime el texto
  delay(1000);
}
```

## Simulación en Proteus

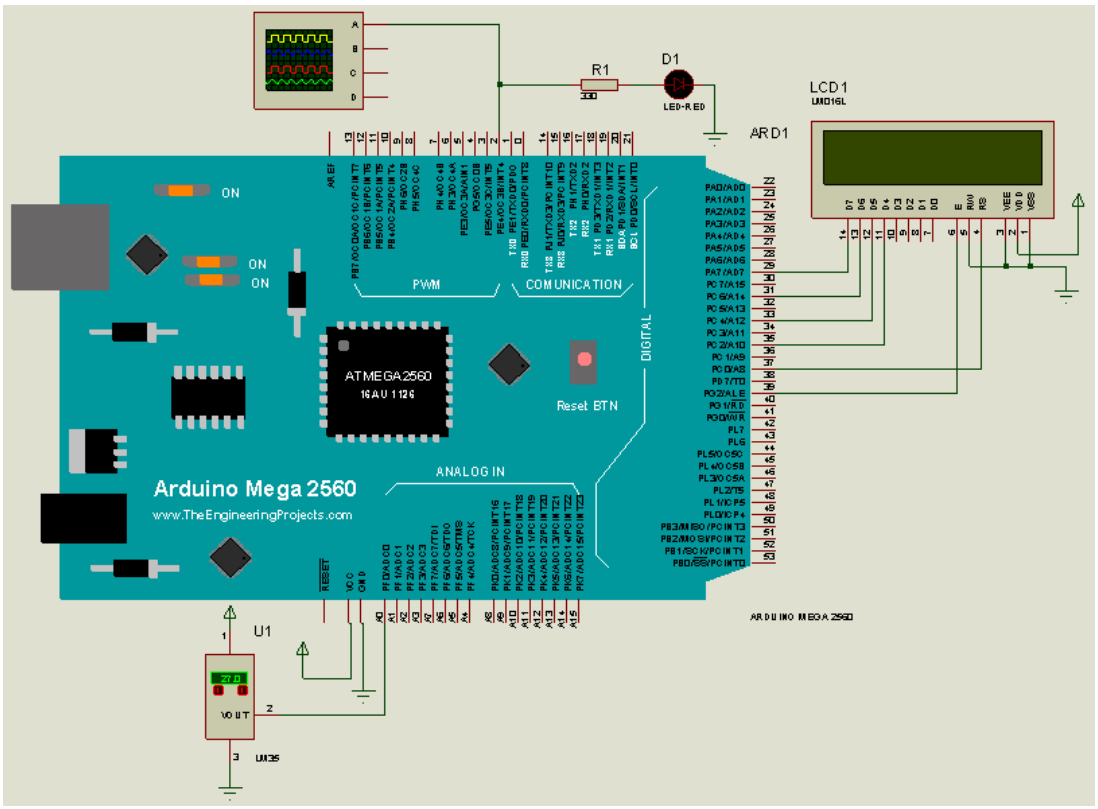


Fig. 5 Simulación de Termómetro Digital con LM35.

## Montaje del Circuito

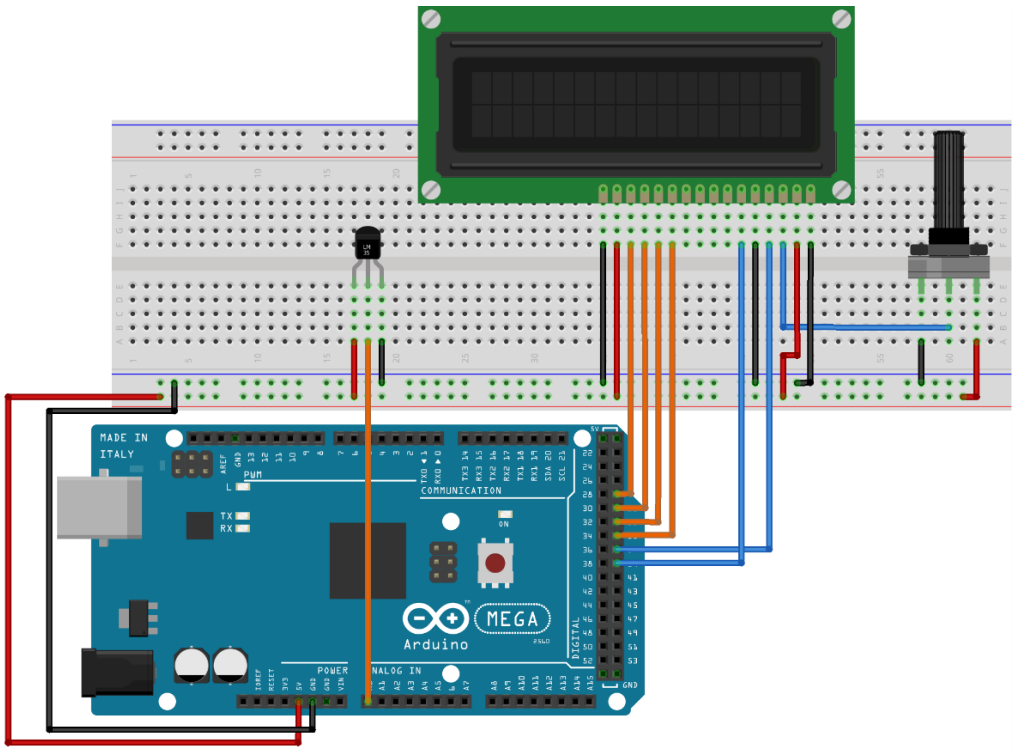


Fig. 6 Diagrama de Montaje del Termómetro Digital.

## Actividad

Del código anterior agregue dos leds uno rojo(caliente) y uno azul(frio), cada uno indica el estado de la temperatura del ambiente. El led rojo se encenderá a partir de los 35°C y el led azul a partir de los 20°C en caso contrario ningún led encenderá.

### 6. Actividades Propuestas

Realizar la Simulación anterior, pero en vez de leds poner dos motores los cuales representarán un aire acondicionado y un calefactor que se encenderán gradualmente a medida que la temperatura cambie según la siguiente tabla:

| Actuador | Duty Cycle (PWM) | | | |
|---|---|---|---|---|
| | 25% | 50% | 75% | 100% |
| Calefactor | 20°C | 15°C | 10°C | 5°C |
| Aire Acondicionado | 35°C | 40°C | 45°C | 50°C |