**Introduction:**

PWM (Pulse Width Modulation) is an important peripheral to get a signal with a specific frequency, to control the power provided to any device.

The speed of motor depends on three factors; load, voltage and current. Using PWM to decrease and increase the amount of power provided to the motor, so control increasing and decreasing of the motor speed.

PWM parameters are amplitude, period and duty cycle.

Frequency= waves per second.

Period= seconds per wave =1/f.

Duty cycle= time on/ (time on + time off).

ISR (Interrupt Service Routine) it’s a piece of code executed when a specific event happens disturbs the normal execution sequence of the code.

Servo motor can be controlled using the PWM to generate its frequency and then get the specific angle.

In this report, both previous topics are implemented using AVR ATmega32.

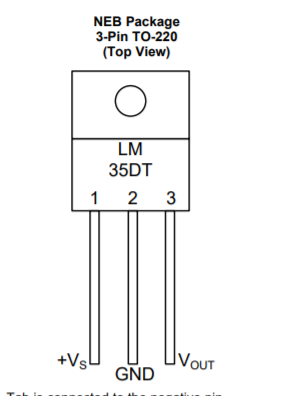
**Temperature controlled DC fan project**

Objective: Display the tempreture value on LCD and control the speed of the DC fan depending on the tempreature sensor readings.

**Components:**

1. LM35 temperature sensor.
2. LCD.
3. L293 motor driver.
4. DC motor.
5. Atmega 32.

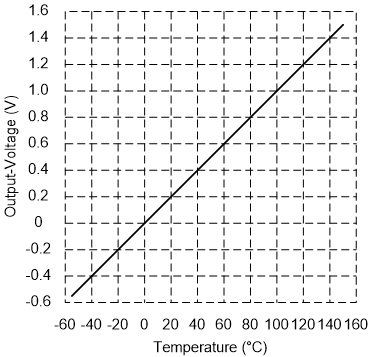
**1)LM35 temperature sensor:**



Source: **LM35 datasheet by Texas instrument.**

It’s an integrated-circuit temperature device, its output volt is linearly proportional with the temperature in Celsius degree.

It’s ranges from −55°C to 150°C.



Its output is 10mv for each degree of Celsius temperature.

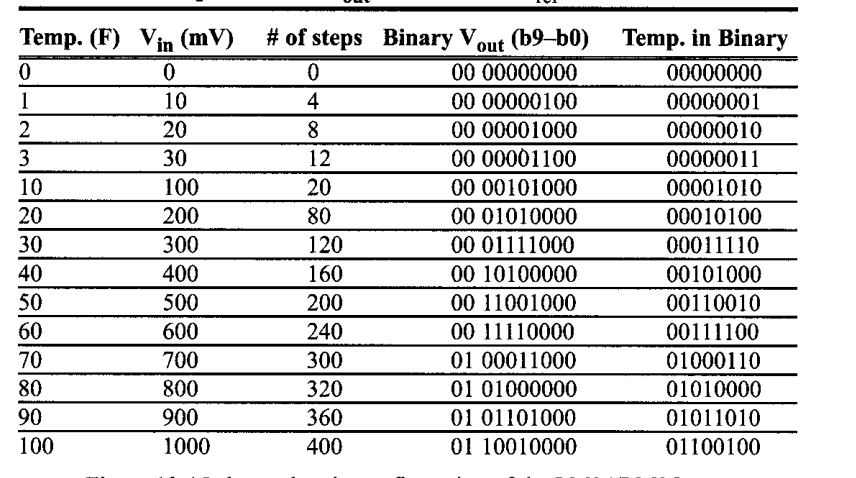
Source: **LM35 datasheet by Texas instrument.**

Output pin is connected to ADC channel 0, with 64 prescaler and internal reference volt which is 2.56 volt.

The ADC resolution is 10 bits with 1024 steps, but LM35 produces 10 mV for each degree, which means that the step size is 10 mV and for full scale, the output volt shall be 10.24 V which is not acceptable.

For 2.56V reference volt, the step size=2.56/1024=2.5 mV. This means the output of ADC will be 4 times the real temperature.

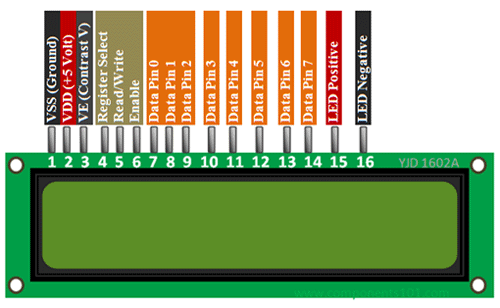
So, the real temperature = ADC read/4, or Use ADCH directly with ADLAR=1.



Source:**The AVR Microcontroller and Embedded Systems: Using Assembly and C, Mazidi.**

**2) LCD:**

Liquid Crystal Display



It consists of LCD panel, LCD controller, CGROM, DDRAM and CGRAM.

DDRAM: The Display Data RAM stores the  
display data represented in 8-bit character  
code.

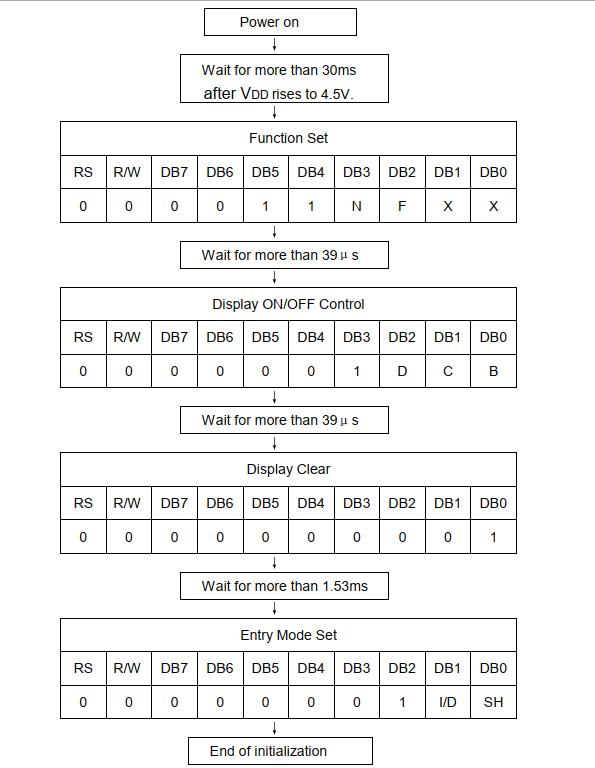
CGROM: saves a predefined pattern for characters.

Source: **https://components101.com/16x2-lcd-pinout-datasheet.**

**Pin Configuration**

|  |  |  |
| --- | --- | --- |
| Pin No | Pin Name | Description |
| 1 | Voss (Ground) | Ground pin connected to system ground |
| 2 | Vdd (+5 Volt) | Powers the LCD with +5V (4.7V – 5.3V) |
| 3 | VE (Contrast V) | Decides the contrast level of display. Grounded to get maximum contrast. |
| 4 | Register Select | Connected to Microcontroller to shift between command/data register |
| 5 | Read/Write | Used to read or write data. Normally grounded to write data to LCD |
| 6 | Enable | Connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement |
| 7 | Data Pin 0 | Data pins 0 to 7 forms a 8-bit data line. They can be connected to Microcontroller to send 8-bit data. |
| 8 | Data Pin 1 |
| 9 | Data Pin 2 |
| 10 | Data Pin 3 |
| 11 | Data Pin 4 |
| 12 | Data Pin 5 |
| 13 | Data Pin 6 |
| 14 | Data Pin 7 |
| 15 | LED Positive | Backlight LED pin positive terminal |
| 16 | LED Negative | Backlight LED pin negative terminal |

Steps of initialization:



Source: **Character LCD datasheet**.

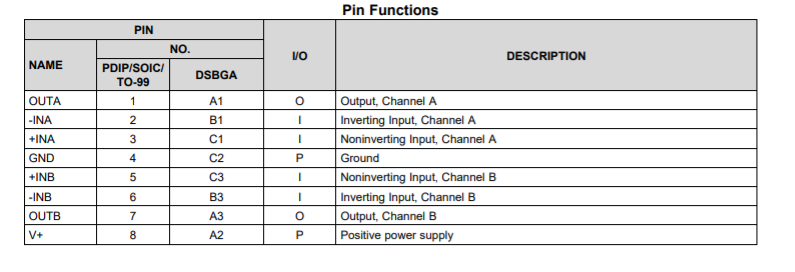
-To send command, Reset RS and RW pins, then send command through data port, then set E pin for 1 ms w.r.t datasheet and then reset it.

-To send data, Set RS and Reset RW pins, send data through data port, then set E pin for 1 ms w.r.t datasheet and then reset it.

**3)L293 motor driver:**

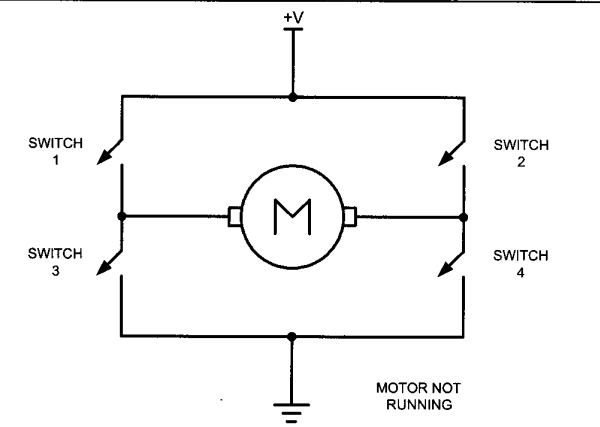
It’s used to control the direction and the speed of the motor depending on the control signal from ATmega32.

The EN1 pin is connected to OC0 to generate a specific PWM signal to control the speed of the motor depending on the temperature value.



Source: **LM293 datasheet, Texas instruments.**

It depends on the H-bridge theory



Source: **The AVR Microcontroller and Embedded Systems: Using Assembly and C, Mazidi.**

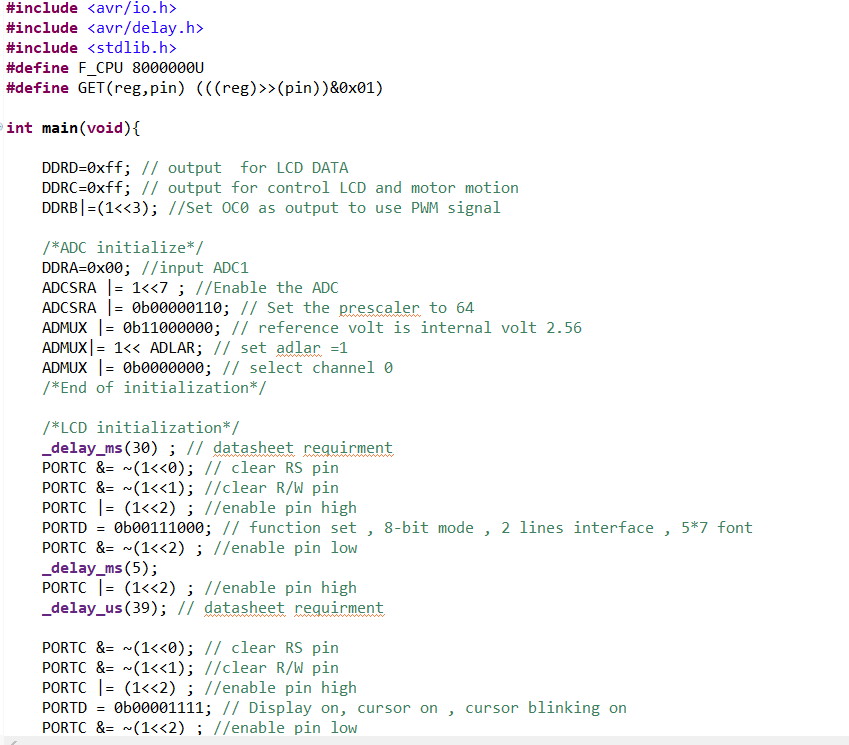
**4)DC motor:**

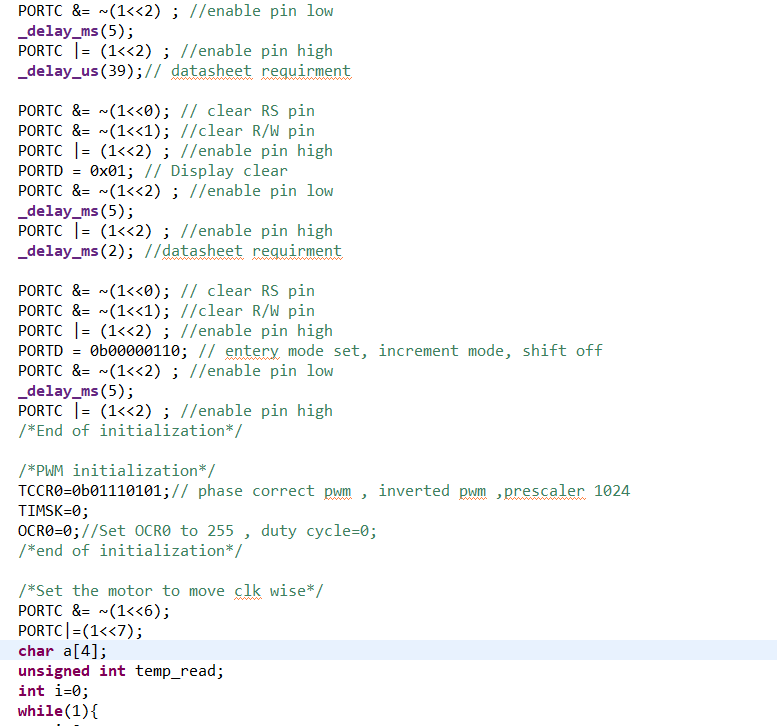
It’s the DC fan representor. It rotates clockwise, with different speed depending on the temperature value.

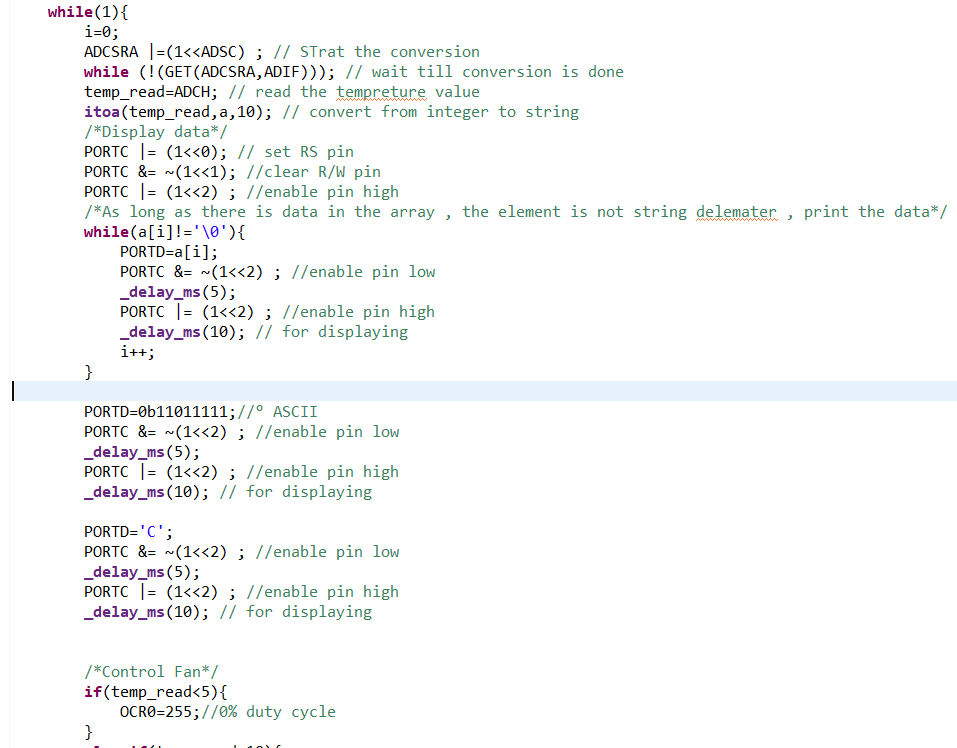
**Pins connections:**

|  |  |  |
| --- | --- | --- |
| Pin | Mode | Description |
| PA0/ADC0 | Input | ADC channel zero, to get the temperature sensor reading. |
| PB3/OC0 | Output | To generate the PWM signal, to control the EN1 pin the motor driver which controls the motor speed. |
| PC0 | Output | Connected to RS pin in LCD |
| PC1 | Output | Connected to R/W pin in LCD |
| PC2 | Output | Connected to E pin in LCD |
| PC6 and PC7 | Output | Connected to IN1 and IN2 in motor driver to control the motor direction. |
| Port D | output | LCD data |

**Code:**









**Configuration:**

ATMega32 frequency=8,000,000 Hz.

**ADC configuration:**

Channel 0, prescaler 64, internal reference volt=2.56 V, ADLAR=1.

**PWM configuration:**

Timer0, phase correct PWM mode, inverted PWM, prescaler 1024.

**LCD configuration:**

8-bit mode, 2 lines interface, 5\*7-pixel, cursor on and blinking, starts from position zero.

**Code description:**

1)Initialize all the peripheral with the previous configuration, and LCD with the datasheet initialization steps by sending commands.

2) start the ADC conversion, wait until it finishes.

3) Get the ADC reading shifted by 2, to convert it to the temperature real value.

4) convert the temperature value from integer to string to display it on the LCD, using itoa function, the first parameter is the integer, the second is the array to put the string on it, the third is the base.

5) loop on the array which contains the string of the temperature, as long as the string doesn’t finish, print the next character on the LCD.

6)Print the (°) on the LCD by writing its ASCII value on the data port.

7)Print the C character.

8) Control the speed of the motor depending on the temperature reading:

a) if the temperature < 5 then don’t open the motor by setting the duty cycle=0, by writing on OCR0=255.

b) if the temperature < 10 then open the motor with only 10% duty cycle, OCR0= 255\*0.9=229.

c) if the temperature < 15 then open the motor with only 30% duty cycle, OCR0= 255\*0.7=178.

d) if the temperature < 20 then rotate the motor with speed 60% duty cycle, OCR0= 255\*0.4=102.

e) if the temperature < 25 then rotate the motor with speed 90% duty cycle, OCR0= 255\*0.1=25.

f) if the temperature is higher than 24 then open the motor with the full speed by making 100% duty cycle, writing OCR0=0.

9) wait for some milliseconds and then Clear the LCD by sending clear command.

And repeat all again.

**Conclusion:**

PWM signals can be used to control the speed of the motor through the motor driver.

Most of the physical quantities are analog, so ADC peripheral is very important to deal with those quantities.

Interrupts have the highest priorities in the code.

**References:**

1. LM35 Precision Centigrade Temperature Sensors datasheet by Texas instrument.
2. Character LCD datasheet.
3. LM293 datasheet by Texas instrument.
4. The AVR Microcontroller and Embedded Systems: Using Assembly and C, Mazidi.