**Car Display**

Embedded C project using AVR-based MCU to measure the current temperature using an LM35 analog sensor, adjust upper and lower temperature thresholds using buttons, and display all the info and warnings using 2 LEDs.

# User Story

Car heating systems is a crucial part of the embedded system of a car as it controls the AC and windows of the car automatically according to the temperature thresholds set by the user. In this project, we simulate the car heating system through:

* Measuring the temperature constantly
* Providing buttons for the user to adjust the upper and lower temerature limits.
* Lighting up upper and lower threshold LEDs gradually to indicate that the current temperature is outside the normal range.
* Displaying the temperature, its status (high, low, or normal), and the upper and lower thresholds on the car screen (LCD).

# Program Flow

1. Initialization

* Initialize the ADC, LCD, PWM libraries.
* Define the PWM pins of the LEDs (3 and 11) as output.
* Write the template of the LCD display.
* Initialize timer 1 in the default counter mood.

1. Main Loop

We use the polling method in order to do the following three tasks sequentially:

* **Buttons:** We check which button the user has pressed, if any, using an ADC interface and we adjust the value of the upper or lower threshold accordingly after waiting for a short time to prevent the bouncing effect of the buttons. We take care not to allow the user to increase the lower threshold higher than the upper threshold or decrease the upper threshold to lower than the lower threshold. We then print the updated counters on the LCD display at their designated positions.
* **LM35 ADC interface:** using the ADC on channel 0, we read the value of the current temperature using the analog LM35 sensor. To avoid reading it every loop, we use timer 1 of the arduino on counter mode to only read it every 6400 ticks.
* **LCD and LEDs:** Through the temperature feedback of the LM35 sensor, we display the temperature on the LCD at its designated position. We compare the temperature to the upper and lower thresholds.
  + Normal: if the current temperature is within range. We turn off the leds and display "Normal" on the LCD.
  + High: if the current temperature is higher than the upper threshold, we light the red LED gradually using PWM interface according to the difference between the temperature and the upper threshold, and we display "High" on the LCD.
  + Low: if the current temperature is lower than the lower threshold, we light the green LED gradually using PWM interface according to the difference between the temperature and the lower threshold, and we display "Low" on the LCD.

# Testcases

2 types of testing were conducted at different stages of the hardware implementation of the project:

* **Whitebox testing:** Each software and hardware component was tested separately to ensure that the end-system would have no underlying bugs.

**The software components being tested were the following:**

* + **Dio Library:** through trying all the Arduino uno pins as both input and output using a dummy led and pushbutton to write and read, respectively, using the library functions with different ways to call it. For example, we once called SetPinStatus(PORTD, 7, 0) and another time we tested SetPinStatus(PORTD, 7, 1).
  + **PWM library:** through testing a dummy led on different duty cycles (0, 40, 100) on all the PWM pins of the UNO.
  + **ADC library:** by trying to read different sensors on different channels at the same time. We found out that the accuracy of the ADC is better when only one sensor is connected (or the mux doesn't have to change channels very frequently). Therefore, We edited the library so as to close the ADC and re-initialize before each conversion. This enhanced the accuracy of the conversion, compromising speed.
  + **LCD library:** through printing different strings on all the possible positions on the LCD.

**The hardware component testing included:**

* + **Pushbuttons:** We took the analog voltage readings of the voltage divider of the pushbutton system and confirmed the conditions using a dummy LED with PWM.
  + **Led system:** different values for resistance were used to make sure the Arduino peripherals are not damages while preserving sufficient brightness level for the LEDs.
  + **LM35 system:** First tested by changing the temperature and noticing the amount of change to the voltage read. Also tested by not changing the temperature and checking if the readings will stay roughly the same.
  + **LCD:** by trying to print fixed sentences on the LCD such as "Hello World".
* **Blackbox testing:** The system was tested as a whole to check if the expected behavior was achieved.
  + **Temperature sensor:** by placing our hand on the sensor and watching the value on LCD as it goes up and down.
  + **Buttons:** by pressing the buttons and noticing the respective counters increase and decrease.
  + **Abnormal conditions:**
    - by increasing the lower threshold to more than the current temperature and noticing the LCD display "LOW" and the green LED increase gradually in intensity.
    - by decreasing the upper threshold to less than the current temperature and noticing the LCD display "High" and the red LED increase gradually in intensity

# BoM

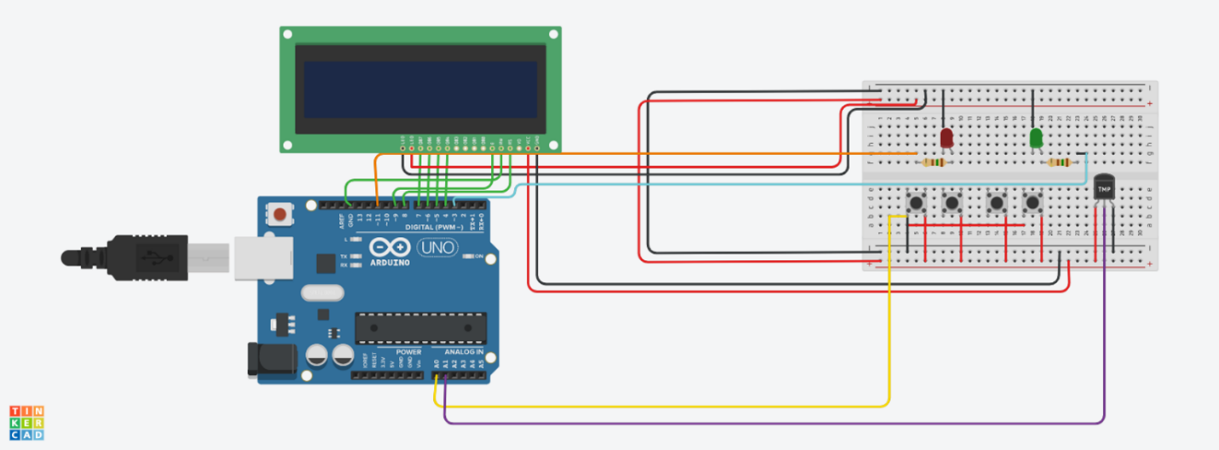
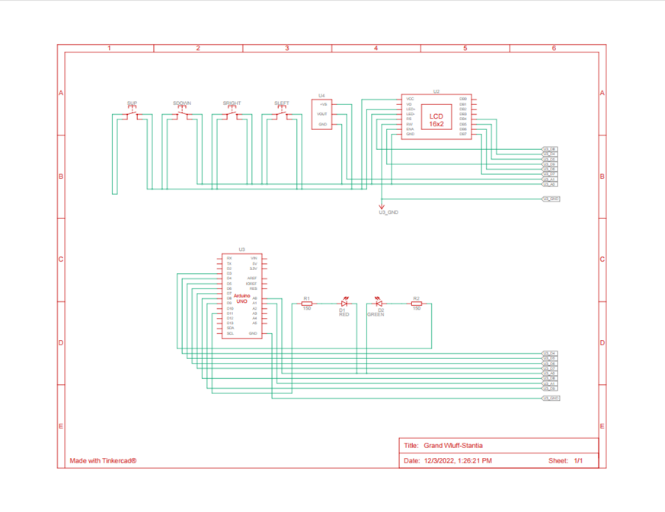
Total Cost: 287 EGP

|  |  |  |
| --- | --- | --- |
| Component | Price in EGP per unit | Total price in EGP |
| Arduino Uno | 125 | 125 |
| Breadboard | 20 | 20 |
| Jumper wires | 0.75 | 15 |
| Resistors | 0.25 | 1 |
| LCD Shield | 100 | 100 |
| LEDs | 0.5 | 1 |
| LM35 | 25 | 25 |

# Video Demo

The video demo can be found at: <https://elsewedyedu1-my.sharepoint.com/:v:/g/personal/yr00067_tkh_edu_eg/EfBeLylbVJNIgqDP3gdSD80Ba01a8p4Pga4nBgPon4ArYw?e=ngTakl>

# Schematic diagram



# Software Design Architecture

Diagram

Description automatically generated

# GitHub Repository

https://github.com/YaraShahin/CarDisplay/

# Contributors

Amr Tarek

Youssef Mokhtar

Yara Shahin