PERFOBOARD PROTOTYPE

Done by Mahmoud Kakouri (7059875), Suraksha Kotte (6996656), Mohammed Zaid (7001897), Michael Madu (6861477)

# 1. Feedback on Deliverable 4

After presenting our circuits for deliverable 4, we received some feedback from our professor as well as from our lab instructor. The main feedback we received is related to deliverable 3 as well, as it stated that our state machine circuit was incorrect and that is due to the MULTISIM circuit in deliverable 3. Our original state circuit had 7 states and based on the feedback we received, we realized that we had some extra, unnecessary states. We used the valuable feedback we received, as well as the application "boole duesto," to redo our state machine, as shown below:

Diagram

Description automatically generated

Fig 1.1: State Machine

Using 3 states only, Q0 represents the fan completely off, while Q1 represents the fan working at normal speed, and finally, Q2 represents the fan working at full speed, and using this state machine diagram we were able to get the following state machine circuit:

Diagram, schematic

Description automatically generated

Fig 1.2: State Circuit on Boole duesto and MULTISIM

This circuit only included two D flip flops, four AND gates, one of which is a three gate AND, and two OR gates, which implies that implementing it on a simulation tool like "TINKERCAD" would be lot easier because we would only require TWO IC's.

Another feedback we got from our professor was that we should integrate all the circuits in one whole breadboard. We showcased the different circuits on many small breadboards using multiple Arduinos.

We received some good comments on our deliverable; we were able to make all of the circuits function on time and create a working prototype of what our final product would look like, and we also received advise on how we should prepare for the upcoming innovation fair.

# 2. Integrated circuit

If you recall, Deliverable 4 of ECTE250 required us to build the four following circuits: sensors, debouncing circuit, motor circuit, and an LCD circuit, however for this week's deliverable, we were expected to integrate all the mentioned circuits in one circuit on a perfboard.

We were able to do this by first integrating the circuit on a full-sized breadboard, and then using that breadboard as a reference, we were able to make the same circuit on a perfboard. We use perfboards because they are more efficient, and because everything is soldered, it is easier to move around, unlike breadboards where a wire may fall off if you move it.

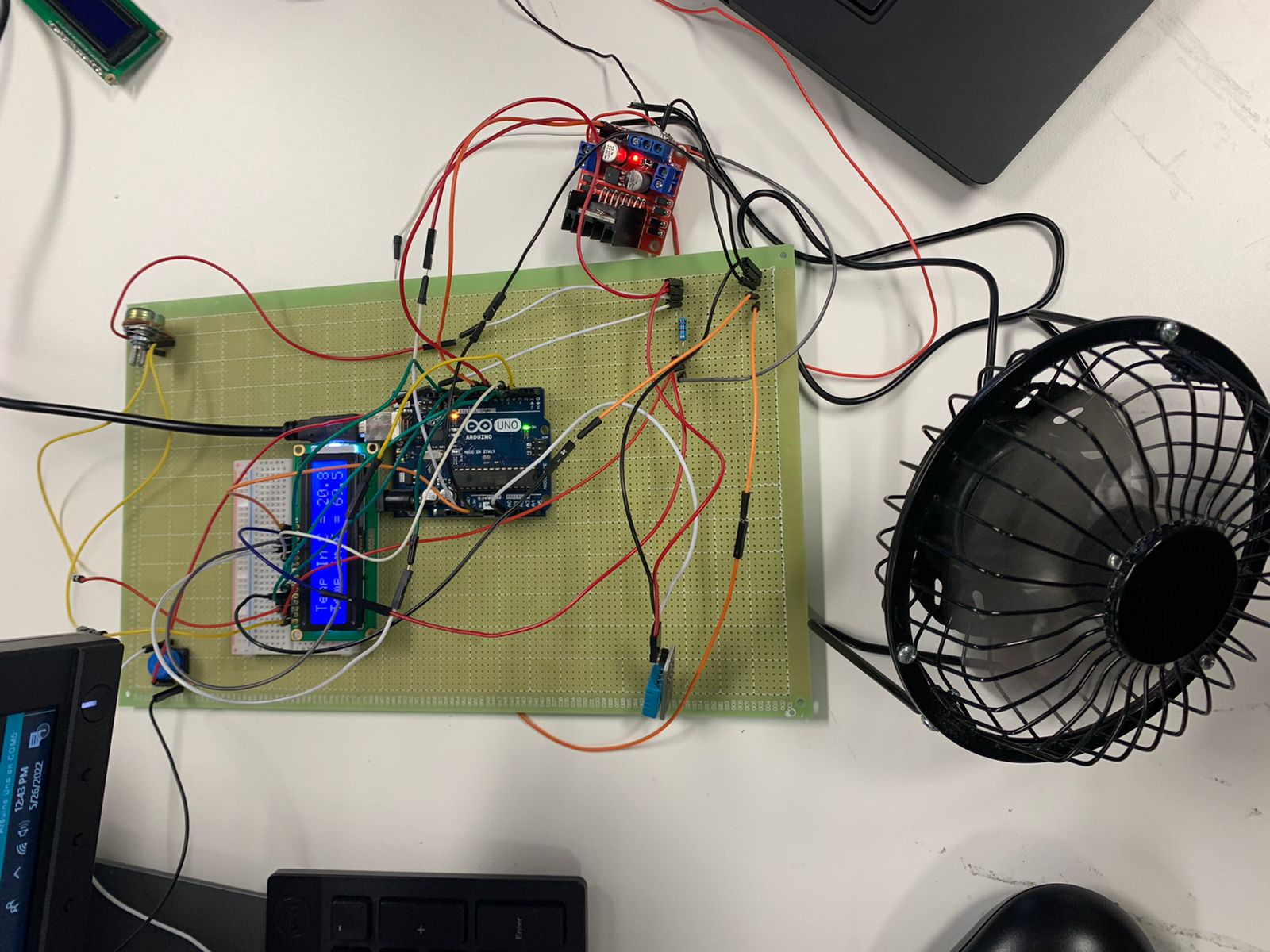


Fig 2.1: Integrated circuit on Perfboard

As you can see in the figure above, we were able to combine all the circuits into one perfboard which displays the temperature and changes the fan’s speed depending on the temperature.

# 3. Connections on the Perfoboard

As shown in Figure 2.1, the LCD is linked to the breadboard, which is then connected to the Arduino uno for simpler connections. To get further into the connectivity, a positive and negative connection are soldered on the side of the perfoboard with red and black wires, respectively. To power the fan, we utilized an L298N motor driver, which was linked to a DC voltage supply. The Arduino's 5V pin has been attached to the right side of the perfboard so whatever component requires connection may be placed beneath the wire and soldered together. The LCD display connectors and the Arduino connections remain the same as in previous deliverables. The main goal of perfboards is to reduce wire connections. Soldering the wires and connections on the back of the Arduino makes it simpler to view. The temperature sensor contains three wires: power, ground, and Arduino. It detects and senses humidity and temperature and displays it on the LCD. The potentiometer connectors regulate the brightness and dimmer of the LCD circuit. As you will know in the following report the blynk app will be virtually connected to the Arduino and you could turn on or off.

# 4. Arduino Code

After combining all the circuits into one perfboard, we had to combine all the different codes for the circuits into one functional code as you can see below:

Text

Description automatically generated with low confidence

Fig 4.1

We started by including all the necessary libraries such as the DHT sensors library and the liquid crystal LCD library, as well as defining the pins for the LCD, motor, and the DHT sensor.

Text

Description automatically generated with medium confidence

Fig 4.2

Graphical user interface

Description automatically generated with low confidence

Fig 4.3

As you can see in the above figure in the void loop section, are the “if statements” where we let the program know what to display and how the fan should operate. When the temperature is is below 16 degrees Celsius the fan will operate in low mode. When the sensor detects the high temperature or above 35 degrees Celsius the fan will go on high mode with fan spinning in full speed. The delays have been added so that the sensor keeps updating the program on what the current temperature is, and the fan works accordingly.

# 5. Blynk Layout

In the feedback, we were told by the instructor to include a graph function or a function that shows the live temperature change feed in the blynk layout along with the power button and the LCD screen which displays the temperature and the humidity to the user. The functionality of the graph is to display the temperature changes and it done live i.e., it updates regularly to display change in temperature. This feature is useful since it allows the user to check the temperature even when they are not present. As previously stated, this device will be used in server rooms, control units, and labs to monitor and maintain the temperature of those rooms, and if there is any sudden change in temperature or humidity, the user or management can discover it early and take measures even if they are not in the area as it works remotely to prevent loss of information, lives, or data while also minimizing/eliminating repair costs and saving time.

Graphical user interface, text, application

Description automatically generated

Fig 5.1: Blynk layout with graph function