Temperature/Humidity and fire emergency automation

1. Theoretical Part

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. [4]

The circuit automates the control of temperature and humidity in rooms and also works as a fire emergency detector.

The temperature and humidity of the air are read by a temperature and humidity sensor named DHT11 which has an operating voltage between 3.5V and 5.5V, a temperature range of 0°C to 50°C and humidity range of 20% to 90% with an accuracy of ±1°C and ±1%.[2].

2. Electrical circuit (Proteus)

The project was implemented using:

- Arduino Uno board
- 1x LCD 16x2 screen
- 7x LEDs
- 3x Relays
- 1x Flame Sensor + flame sensor library [1]
- 1x DHT 11 Temperature and Humidity sensor + library [3]
- 1x 5V DC source
- 1x Buzzer/speaker
- 4x Resistors

The DHT11[3] sensor sends the temperature and humidity data to the Arduino board which reads it through pin A0. Both data are displayed on the LCD screen and if the temperature goes under the minimum temperature, the Arduino board activates the first relay which is a heater connected to a power source (LED used instead of a heater in proteus for obvious reasons). An red-LED connected through the relay coil indicates if the heater is ON or OFF. An OFF red-LED indicates that the heater is ON and an ON LED indicates that the heater is off.

Same principle applies for the humidity of the air as for the temperature.

Also, the Arduino can control a house light.

In case of a flame detected by the Flame Sensor*[1](in proteus the sensor is controlled by its test pin which works like a digital 0 1, 0-no flame detected, 1-flame) the Arduino board

enters the emergency state displaying an appropriate message on the LCD display, shuts off the heater, starts the humidifier and sounds the alarm while turning ON the emergency LED. The reason the humidifier starts is because fire has a hard time burning in humid air. After the flame is eliminated, the Arduino board resets[2] and works like before.

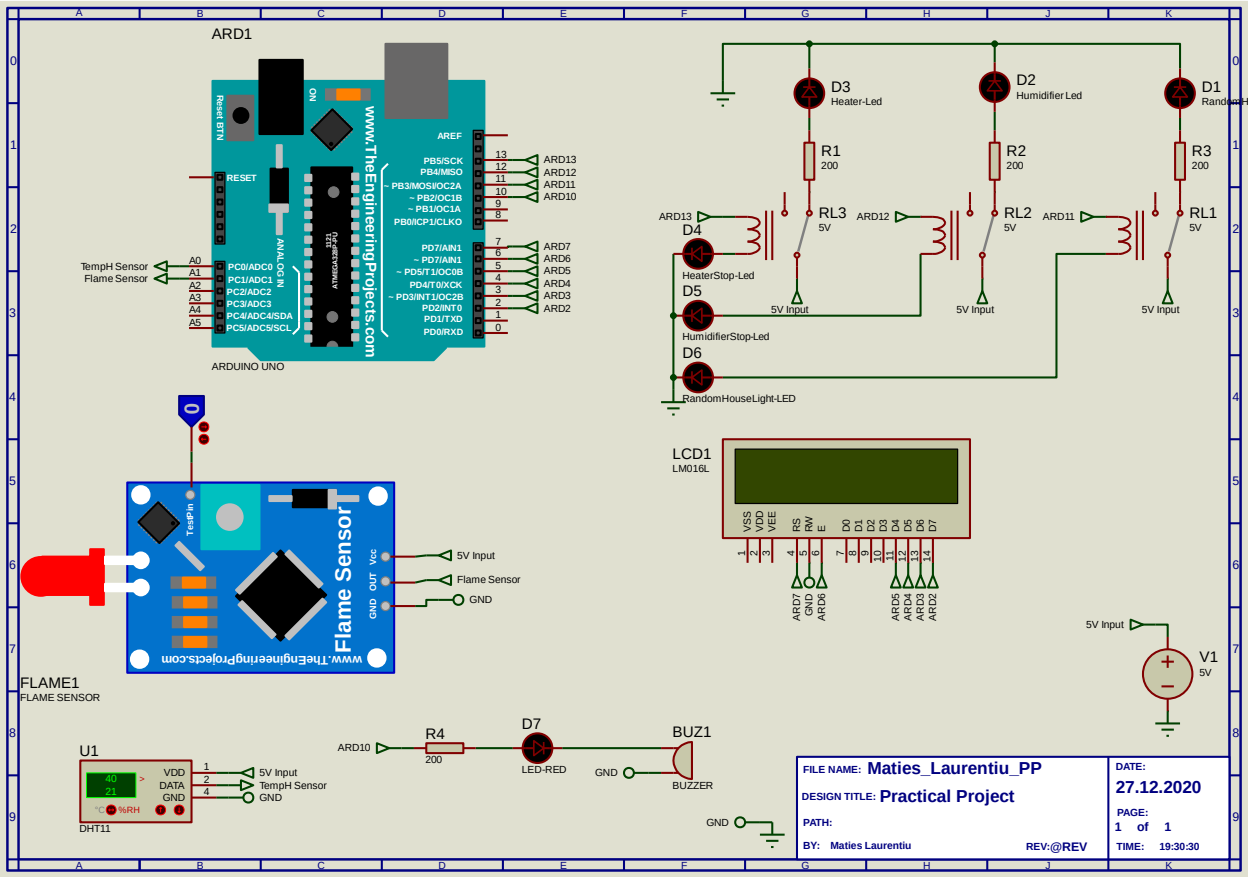

Figure 1. Project electrical scheme

*flame sensor library needs to be downloaded and included in the proteus libraries.

3. Arduino IDE

For the implementation of the Arduino sketch, downloading the DHT library[3] is necessary.

```
#include <LiquidCrystal.h>
#include <DHT.h>

#define DHTPIN A0 //input pin for temp-humi sensor
#define flameSensor A1 //input pin for flame sensor

//declaring the
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void(* resetFunc) (void) = 0;
```

Figure 2. Libraries included and defining sensors and sensor pins

The sketch starts by including the necessary libraries for LCD screen and DHT sensor, defining the sensor pins and declaring the type of DHT sensor used. The void function at the beginning is used to reset the Arduino board without using the physical reset pin(Figure 2).

Next part is declaring the pins used to control the heater, humidifier, house light and fire alarm. Next variables are used to identify different events or to set priorities which we will see in further part of the code. (Figure 3)

```
LiquidCrystal lcd(7,6,5,4,3,2);

int light1 = 13; //heater pin
int light2 = 12; //humidifier pin
int light3 = 11; //house light pin
int buzz = 10; //fire alarm pin

int show = 0; //lcd display cases
int ER = 0; //if fire emergency was triggered
bool reset = false; // for arduino reset if ER is > 1
bool flamePriority =false;// for working in case of fire
char temp[] = "Temp = 00.0 C ";//for displaying Temperature and humidity on display
char humidity[] = "RH =     % ";
```

Figure 3. Pin declaring and other variables

```
void setup() {

  Serial.begin(9600);
  lcd.begin(16,2);
  dht.begin();

  //setting pin modes
  pinMode(light1, OUTPUT);
  pinMode(light2, OUTPUT);
  pinMode(light3, OUTPUT);
  pinMode(buzz, OUTPUT);
```

Figure 4.Setup of the Arduino board

The setup function sets the mode of the pins, reading speed, LCD resolution and starts the reading of the DHT sensor. (Figure 4)

The loop function constantly reads the temperature and humidity of the air then it enters the flame priority IF statement. If this statement returns FALSE, everything works normally but if it returns TRUE then the Arduino will use only the "flameDetect" function which works uniquely if a flame is detected. (Figure 5)

```
void loop() {

  // Read humidity
  float RH = dht.readHumidity();
  //Read temperature in degree Celsius
  float Temp = dht.readTemperature();

  //if flamePriority is false, arduino works normally
  //if flamePriority is true, instructions from flameDetect have priority
  if(flamePriority == false){
  //temperature and humidity conrol
  tempControl(Temp);
  humidityControl(RH);
  //temperature and humidity display
  tempH(RH, Temp);
  }

  //flame detector
  flameDetect();
}
```

Figure 5. Loop function

In figure 6 is the flame detector function which has working priority over the normal working state of the Arduino. If a flame is detected it remembers that the emergency was triggered

```
//flame detector
void flameDetect(){

  //when the flame sensor pin(A1) reads a HIGH value
  //it triggers the alarm, displays a message on the lcd display
  //when the emergency is gone, resets the arduino
  if(digitalRead(flameSensor) == HIGH){

    ER=1;
    lcdShow(1);
    flamePriority = true;
    digitalWrite(light1, HIGH);
    digitalWrite(light2, LOW);
    digitalWrite(buzz, HIGH);
    if(ER == 1){ //if Emergency was triggered at least once, it approves the reset function
      reset = true;
      ER=2;
      }
    }
    else{
      digitalWrite(buzz, LOW);
      flamePriority = false;
      if(ER ==2 && reset == true){ // If one emergency took place and reset is approved, resets the ardu
        resetFunc();
      }
```

Figure 6. Flame detector function

through the use of "ER" variable which is used later in the function along with "reset" variable to control the reset of the board. After displaying the appropriate message on the LCD display and setting the priority of the loop functions, the Arduino will shut down the heater, start the humidifier and sound the alarm along with the LED indicator that a flame was detected.

After all the instruction for the emergency, an IF statement approves the reset of the board, then if the flame sensor doesn't detect the flame, the Arduino stops the alarm, resets the flame priority and then it resets itself to work in normal conditions. If we wouldn't use the "ER" and "reset" variables, the Arduino would reset on every loop, this making the LCD display the message alternatively with the temperature and humidity and the buzzer/speaker sometime working and sometimes not. (Figure 6)

```
//temperature control
void tempControl(float Temp){

    if(Temp <= 21 ){
      digitalWrite(light1, LOW);
    }

    if(Temp >= 23){
        digitalWrite(light1, HIGH);
    }
}
```

Figure 7. Temperature control function

The temperature control function which activates the heater relay if the temperature is above the limit or deactivates is if the temperature is under a limit. (Figure 7)

The humidifier control function works like the temperature one. (Figure 8)

```
//humidifier control
void humidityControl(float RH){
    if(RH <= 35){
      digitalWrite(light2, LOW);
    }

    if(RH >= 50){
        digitalWrite(light2, HIGH);
    }
```

Figure 8. Humidifier control function

The temperature and humidity function is used to display the values on the LCD screen.

```
//temperature and humidity display
void tempH( byte RH, byte Temp){

    delay(1000);           // wait 1s between readings

  // Check if any reads failed and exit early to try again
  if (isnan(RH) || isnan(Temp)) {
    lcdShow(3);
    //lcd.clear();
    //lcd.setCursor(5, 0);
    //lcd.print("Error");
    return;
  }

  temp[7]      = Temp / 10 + 48;
  temp[8]      = Temp % 10 + 48;
  temp[11]     = 223;
  humidity[7]         = RH / 10 + 48;
  humidity[8]         = RH % 10 + 48;
  lcdShow(2);
  //lcd.setCursor(0, 0);
  //lcd.print(temp);
  //lcd.setCursor(0, 1);
  //lcd.print(humidity);
}
```

Figure 9. Temperature and Humidity display function

The IF statement checks if any reads failed or exit early and then displays the “Error” message. If the statement passes, variables “temp” and “humidity” which are string vectors are modified by inserting the values read by the sensor in the string and then displaying it. (Figure 9)

The “lcdShow” function is used to better organize the messages shown on the LCD display and it is called by any function that requires a message displayed. Each case has a number from 1 to 3, “1” shows the fire emergency message, “2” shows the temperature and humidity and “3” a error message if DHT11 sensor couldn’t read values. (Figure 10)

```
//display for different cases
void lcdShow(int s){

  //display if flame detector is triggered
  if(s == 1){
    lcd.setCursor(0,0);
    lcd.print("FIRE          ");
    lcd.setCursor(0,1);
    lcd.print("EMERGENCY!");
    } else if(s == 2){    //display temperature and humidity
      lcd.setCursor(0, 0);
      lcd.print(temp);
      lcd.setCursor(0, 1);
      lcd.print(humidity);
    } else if(s == 3){    //display error message if temperature and humidity couldn't be read
      lcd.clear();
      lcd.setCursor(5, 0);
      lcd.print("Error");
    }
}
```

Figure 10. LCD display cases

Reference list

[1] Flame Sensor Library for Proteus - https://www.theengineeringprojects.com/2016/07/flame-sensor-library-proteus.html

[2] Function to reset the Arduino board - https://www.theengineeringprojects.com/2015/11/reset-arduino-programmatically.html

[3] DHT sensor library - https://www.arduino.cc/reference/en/libraries/dht-sensor-library/

[4] https://www.arduino.cc/en/guide/introduction