IncuBaby (IB04) Design of a Low-End Controller for a Premature Infant Incubator Connected to Node-RED on a PC for High-End User Interfacing Through the Web Dashboard

Eastern Mediterranean University

CMPE320 Embedded System Design

Team-5 Final Report Revision 1

Date of Submission: **21/06/2023**

TEAM LEADER:

5.1 Berke Emerce 20700028

TEAM MEMBERS:

5.2 Abdulmalik Olaoye Raji 18701340

5.3 Hazal Aydın 18330739

5.4 Halil Yüksel 20000134

5.5 Buse Alasköz 20000049

5.6 Cahit Osmanlı 20331145

5.7 Aydın Eğitmen 19331193

5.8 Servet Nursu Boztaş 17330154

Ethical Declaration:

“We are aware of the Principles of Research Ethics as should be obeyed and we declare

that this report which we submit to the instructor will be the result of our own

independent work and that in all cases, material from the work of others will be fully

cited and referenced as required by the academic rules and ethical conduct. We

understand that if any kind of plagiarism is detected in our written work, the instructor

will set our whole homework-project grade to zero, and take the case to the ‘Disciplinary

Committee’ for necessary action.”

**CONTENTS**

[**Design Objectives** 3](#_Toc138191718)

[**Overall System** 3](#_Toc138191719)

[**Introduction** 4](#_Toc138191720)

[**Social Impacts** 4](#_Toc138191721)

[**Design Requirements** 5](#_Toc138191722)

[**Modeling and Analysis for Conceptual Design** 7](#_Toc138191723)

[***Part A: Simulation of the Physical System*** 7](#_Toc138191724)

[***Part B: Simulation of the Whole System*** 9](#_Toc138191725)

[**Design of the Low-End Controller** 11](#_Toc138191726)

[***Arduino Internal LED*** 11](#_Toc138191727)

[***Sensor Interfacing and ADC*** 13](#_Toc138191728)

[***UART Received Character Processing*** 15](#_Toc138191729)

[***Control Task and PWM Operation*** 16](#_Toc138191730)

[***Design of the LCD*** 19](#_Toc138191731)

[***Design of the Alarm, UART Data Transmission and Testing*** 20](#_Toc138191732)

[***Code for Setup and Loop*** 22](#_Toc138191733)

[**Node-RED Implementation** 23](#_Toc138191734)

[***Configuring Proteus*** 23](#_Toc138191735)

[***Configuring VSPE*** 23](#_Toc138191736)

[***Node-RED software*** 24](#_Toc138191737)

[**Conclusion** 26](#_Toc138191738)

[**References** 29](#_Toc138191739)

# **Design Objectives**

(i-Halil, w-Abdulmalik)

The objective of this project is to design and emulate a baby incubator that has the ability to keep a prematurely born infant in a safe temperature range. To achieve this, two temperatures, namely the air temperature and the baby surface temperature have to be taken as physical measurements. These temperatures must be controlled using a cyber physical system. Neither the air temperature nor the baby surface temperature is allowed to go above temperatures where it would be harmful for the infant. A desired temperature is set as well as a proportional gain which allows the system to make a set of calculations to set the duty-on time of a heater that is present in the incubator allowing a safe temperature to be kept inside of the incubator chamber. The cyber-physical simulation is first built on the SciLab software and the conceptual interactions are tested. Then the low-end controller is built in the Proteus software utilizing an Arduino Uno which shows necessary values using an LCD display. Finally, the data from the controller is visualized using the Node-RED software running on a PC.

# **Overall System**

(i-Berke, d-Buse, t-Halil, w-Berke)

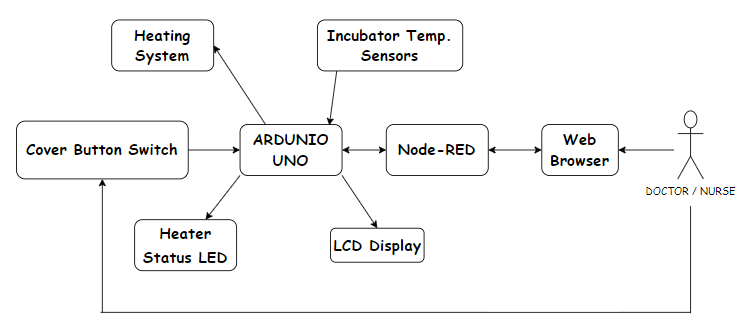


Figure 1 - Overall System Diagram

The overall system consists of three parts: The physical components, the low-end controller and the high-end Node-RED software running on a PC. The thermistors obtain the temperature values ThA and ThB referring to air temperature and baby surface temperature respectively. These readings are fed into the low-end controller. The two NTC sensors are connected to the AD0 and AD1 pins of the Arduino along with two resistors. As for other components we have a virtual terminal connected to pins IO1 and IO2, an LED connected to the IO12 pin, an LCD display connected to IO pins 2-7, a switch controlling the button for the incubator cover as well as a COMPIM which models a physical serial port that is connected to the same pins as the terminal. Sketch code is run on the Arduino to make calculations which allow us to determine the temperature readings in a correct format, whether the heater shall be on or off, the power percentage and whether there should be an alarm state ongoing. The air temperature (ThA), the baby surface temperature (ThB), the desired temperature (ThD), the proportional gain, the alarm state, power percentage and cover status are shown on the terminal which, through COMPIM, is sent to the Node-RED software. To achieve the transfer of data the Virtual Serial Port Emulator (VSPE) is used. The Node-RED user interface is used to program the functions which allow us to show the same values we previously showed using the terminal on a more user-friendly dashboard panel of the Node-RED software. The linkage of these components is shown in Figure 1.

# **Introduction**

(i-Halil, w-Halil)

This report will present the total process from the design, the prototyping and the testing of the IncuBaby (IB04). Throughout the project, agile development and rapid prototyping practice has been implemented. Each incremental step that got our team to the end design has been documented and will be presented. The big picture in this picture is of great importance as it directly relates to the well-being of a compromised newborn. Therefore, testing such a design in the best way is of utmost importance.

# **Social Impacts**

(i-Hazal, w-Hazal)

The design of the DigiTerm-04 IncuBaby IB04 incubator has significant social impacts in the field of infant healthcare. By accurately controlling the infant body temperature, it ensures an optimal environment for babies in incubators, promoting their growth and well-being. The system prevents temperatures above 37°C, which are known to be harmful, by triggering an alarm state if the air or body temperatures are exceeded. This technology plays a crucial role in reducing mortality rates and improving overall health of infants. In terms of potential health hazards, this design does not indicate the presence of radioactivity or other harmful agents. The components used such as NTC thermistors and Arduino-Uno, are standard electronic elements commonly used in various systems. Therefore, there is no inherent risk of exposure to radioactivity or substances that may cause health problems. The focus of the design is primarily on precise temperature regulation and control, ensuring the safety and well-being of infants without introducing any additional health concerns. There is a lack of testing done on the safety of using Arduino Uno chips in medical instruments however so until sufficient safety evidence is provided, it shall only be used for prototyping purposes.

**Design Resources**

(i-Cahit, w-Cahit)

We used the following resources during our project:

* Personal computers to do online meetings and use the software needed for our project as well as for research.
* 7 team members and a project leader.
* Design Algorithms from laboratory manuals.
* Scilab 2023.0.0 software for the design and simulation of the cyber-physical system.
* Proteus 8 software, virtual oscilloscope, virtual terminal, two NTC sensors, LCD display, resistors, LED diode, virtual Arduino UNO board, COMPIM.
* VSPE software for the emulation of the serial ports.
* Node-RED software for the high-end modelling and dashboard view.

# **Design Requirements**

(w-Cahit)

The design requirements for the DigiTerm-04 are taken from “Technical Requirements for IB04 Incubator Project” (M. Bodur). [1].

a. it must be developed on an Arduino-Uno r3 board (16MHz clock).

b. It must have a blink-alive LED (internal-LED of Arduino) flashing for 100ms once at every two seconds.

c. Sensors must be two 4k7 NTC thermistors (for temperatures ThA and ThB) with a suitable linearization resistor to read temperatures between 20 … 45 degrees with less than 0.5 oC accuracy. 4k7B3900NTC sensor is a temperature dependent resistance.

* AD conversion of both NTC sensors (ThA and ThB) should be once at every 50 seconds. Readings of ThA and ThB should be stored as 10-fold integers to have 0.1 C precision (i.e., physical 𝑇𝐴=32.6 C shall be stored as integer ThA=326).
* The precision of the NTC sensor is 0.2 C in the range 0 - 70 C when calculated by the expression 1/𝑇 = 1/𝑇𝑜 + (1/𝐵) 𝑙𝑛(𝑅𝑡/𝑅𝑜) using 𝑇 in Kelvin (𝑜K).
* Your report shall clearly show that your ADC has less quantization error than your thermistor sensor circuit.
* Best linearization for the 20…50°Crange is obtained using 𝑅𝑠 = 2700 kΩ in sensor circuit. ADC configuration should provide suﬀicient dynamic quantization ratio to read the sensor in full dynamic range ratio for the temperature range 20 - 50 °C.

d. It should set the parameters Kp, and ThD by the UART command (#)p , and (#)d ; where # denotes a decimal integer number. For example (15)p shall set Kp=15.

* For these tasks, use a temporary integer N for parsing the numeric value. If the received ASCII character C is '(', that is C=40, transition on this guard should accomplish a set action N=0. When received character code satisfies 48 C 57, transition should accomplish a set actions N=N\*10+(C–48).

e. Control Task

Control action requires proportional feedback gain Kp. At every minute it shall do the following tasks:

* measure both temperatures ThA and ThB. You can set the temperature of a thermistor directly by using + and - buttons on its circuit symbol.
* calculate e = ThB – ThD
* calculate percent-Power, PP = Kp \*e.
* PWM period shall be 50 s. Duty time will be Td= 50\*PP/100=PP/2 seconds
* keep the heater output high for Td seconds.

f. Display these values in a clearly recognizable format on the 16x2 LCD display:

* Temperatures ThA and ThB (integer 10\*temperature in C)
* ThD (desired-ThB) temperature (integer, {320, 340, 360}, represents {32, 34, 36}C with one digit after decimal point.)
* Proportional gain Kp (integer)
* Alarm status, for air temperature ThB>37, make AA=1, else AA=0,
* You can print different variables on the screen in odd seconds and even seconds so that the LCD screen can be read more easily.

g. The UART shall transmit:

* Temperatures ThA, ThB, ThD
* Proportional gain Kp,
* Button SwCov (for cover status, low: cover closed, high: cover open),
* Alarm status AA (to display graphically the status of the incubator on the Node RED dashboard through transferring data to a PC).

h. Node-Red (on PC) and its web dashboard should provide following properties:

i. it should provide access to the 9600 Baud, 8 data, no parity, one stop virtual serial port at the Prosis Arduino simulation.

1. receive ThA and ThB: temperatures, PP: (%Power) PWM level percentage, and AL: alarm status, and display them on the dashboard in a convenient format.

ii. construct a Node-RED dashboard

a) to display plot of temperatures ThA and ThB, and heating PWM power percentage PP,

1. to set controller setting Kp, and desired temperature ThD in a conveniently and user-friendly manner.
2. to display the status of cover, and alarm condition on the dashboard.

# **Modeling and Analysis for Conceptual Design**

(i-Hazal, d-Servet, t-Abdulmalik, w-Berke)

The first step of our design is the simulation of the cyber-physical system using the SciLab 2023.0.0 software. The XCOS graphical editor found in the Scilab software was used to model the system. The components needed to model the system were picked from the palette browser and the settings of these components were adjusted to provide a working model for our conceptional system.

## ***Part A: Simulation of the Physical System***

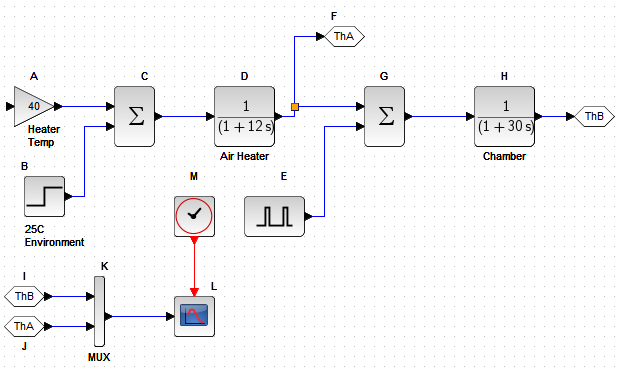


Figure 2 - Physical Part of the Simulation

The following steps were used in the physical modelling:

1. A gain block was placed to simulate the heater heating up the air by 40oC.
   1. Gain: 40
2. A step function was put to simulate the environment temperature as 25.
   1. Step Time: 0
   2. Initial Value: 0
   3. Final Value: 80
3. A summation block to add up the effect of the gain block to the environment temperature.
   1. Datatype: 1 (real double)
   2. Inputs: [1;1]
   3. Do on Overflow: 0 (nothing)
4. A CLR block to simulate the air heater.
   1. Numerator: 1
   2. Denominator: (1+12\*s)
5. A GOTO block to output ThA.
6. A PULSE\_SC to show the square wave generator that simulates the cover being opened for 20 seconds at the 200 second mark where the temperature drops by 10 degrees Celsius.
   1. Phase delay: 200
   2. Pulse Width: 5
   3. Period: 400
   4. Amplitude: -10
7. Another summation block to add the effect of the square wave generator.
8. Another clear block for the chamber.
   1. Numerator: 1
   2. Denominator: (1+30\*s)
9. GOTO block to output ThB.
10. A CSCOPE block to show the outputs on a graph
    1. Ymin: 0
    2. Ymax: 70
    3. Refresh period: 400
    4. Buffer size: 400

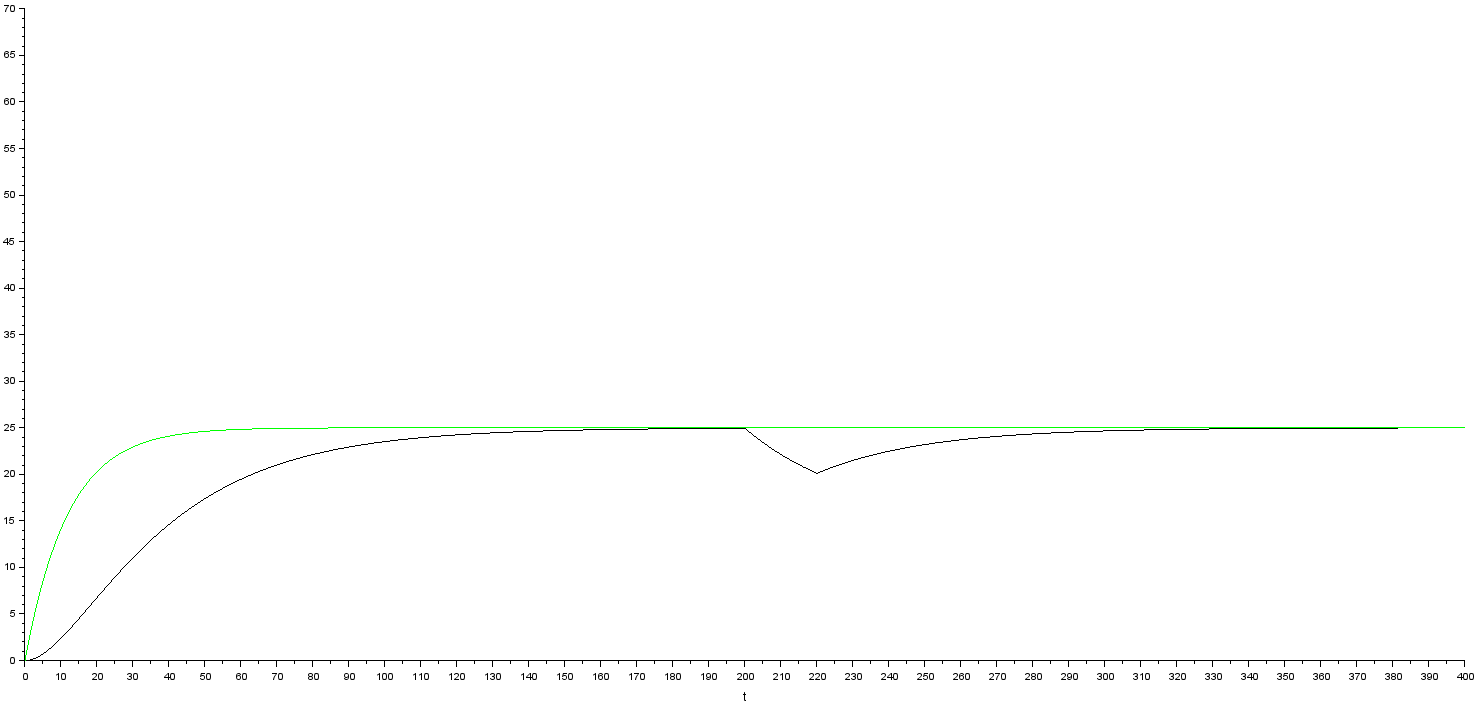


Figure 3 - Results of the simulation of the physical system

After running the simulation, when the physical part is running stand-alone, we obtain a graph where we can see the convergence to the step function value (25 degrees Celsius) as well as the temperature drop effect at 200 seconds lasting for 20 seconds which is what was intended with the design. This is clear to see from Figure 3.

With the expected results of the simulation being met, we move on to adding the cyber part to the physical part.

## ***Part B: Simulation of the Whole System***

(i-Nursu, d-Hazal, t-Abdulmalik, w-Aydın)

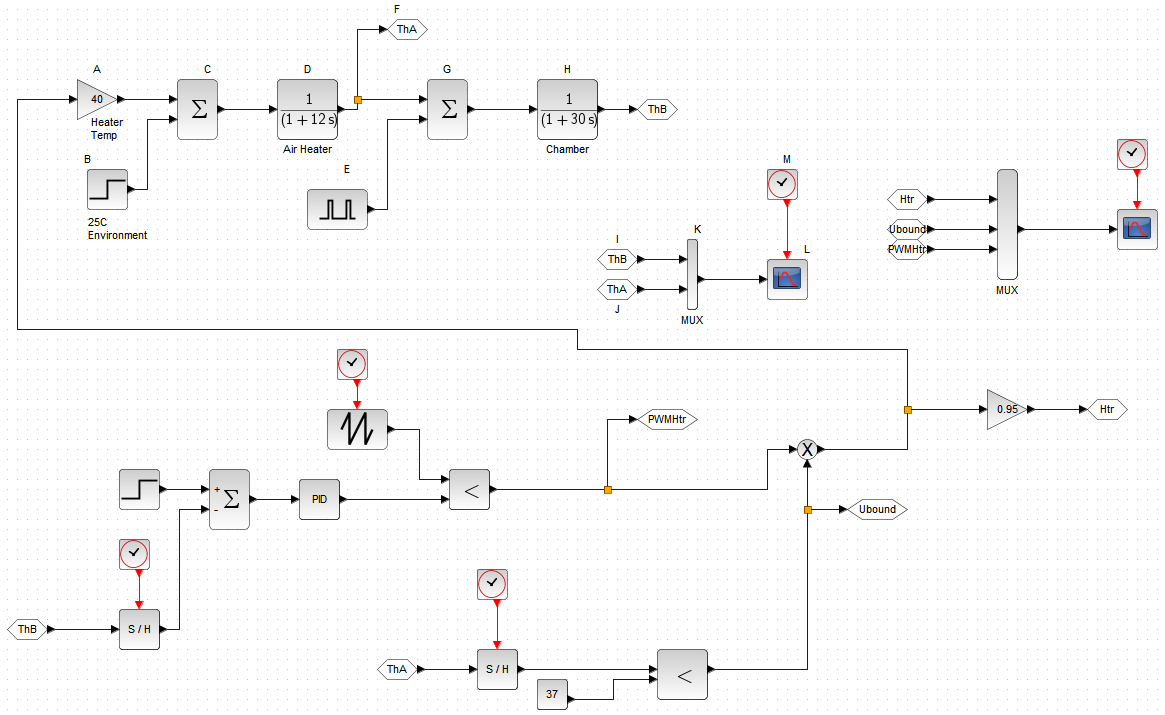


Figure 4 - The simulation for the entire system

The following steps were used in the physical modelling:

1. Sample Hold for ADC of ThA and ThB have a sampling period of 1 minute. Parameters of the blocks are:
   1. Samphold for ThB with datatype: 1
   2. CLOCK\_c for ThA samphold: 1,0
2. PID controller is composed with a step input function for ThD, and the sampled ThB value:
   1. STEP\_FUNCTION for ThD: 0, 0, 32
   2. SUMMATION block for PID: 1, [1; -1],0
3. A sawtooth function was placed after PID in order to simulate a PWM for PID output:
   1. PID: Proportional 10, Integral and Derivative 0 [1,0,0]
   2. Sawtooth\_f for PWM: no parameter.
   3. CLOCK\_c for ThA samphold: 1, 0.
   4. RELATIONALOP operation for “< “: no zero-cross, data type: 1 (double).
4. In order to provide upper bound restriction for temperatures above 37C, we use a Logical AND operator and compare sampled value of ThA by 37C, and multiply the result to PWM output. Parameters of the blocks are:
   1. CONST for Upper Bound Value: 37
   2. RELATIONALOP operation: no zero-cross, datatype: 1 (double)`
   3. GAINBLK: 0.95, It shifts Htr curve on CSCOPE for better visibility.
   4. MUX for ThA-ThB CSCOPE: 2 (two inputs)
   5. CSCOPE for Htr-UBound: Ymin, Ymax, Refresh, Buffer: 0, 1, 400, 400.
   6. CLOCK\_c for ThA-ThB cscope: 0.01, 0.

The overall system consists of the cyber part and the physical part being connected together. We run the temperature readings through the ADC. The error is found for ThB and a proportional gain is applied for the PP calculation. We compare it with out PWM counter to find the duty-on-time. We compare the ThA value with 37 degrees Celsius and these two parameters go into the heater function we previously put in the physical part.

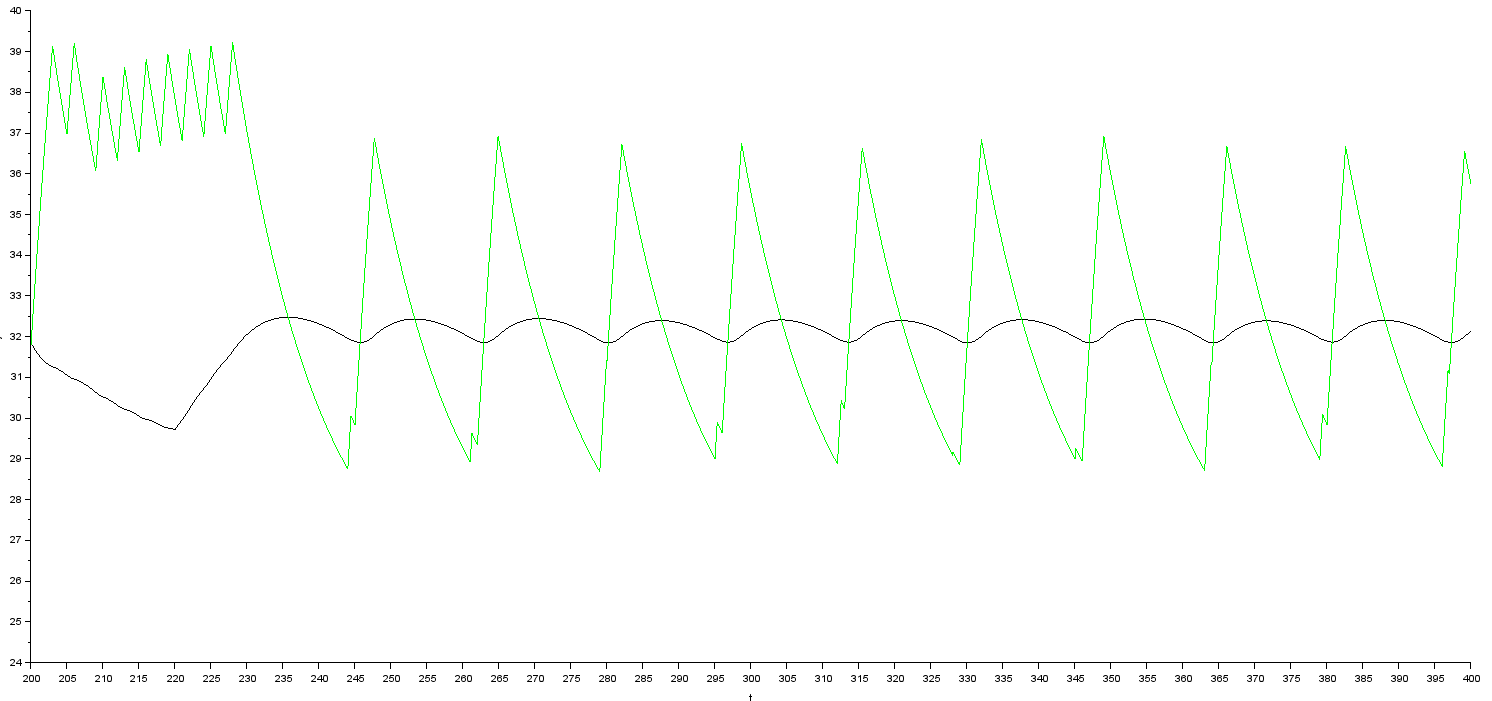


Figure 5 - Simulation of the entire system with Kp=10

Upon running the simulation, we obtain a graph. After the overshoot we see a pattern of oscillations form. The system can be defined as stable in the sense that there is a recurring oscillation pattern and unstable in the sense that it doesn’t converge to a steady state value. We obtained the results in Figure 5 using proportional control value of 10. The result may be improved upon lowering the proportional value.

From Figure 6, It is easy to see the vast effect of Kp. Lowering it down to a value of 2 which is the lowest meaningful whole number value (a value of 1 being just the error itself), we get a much more stable graph. The system mostly stabilizes more better than with Kp=10. We see lower amplitude oscillations and at the end even though a totally steady state is not reached, the maximum values of the oscillations don’t go much above the desired temperature of 32 degrees Celsius.

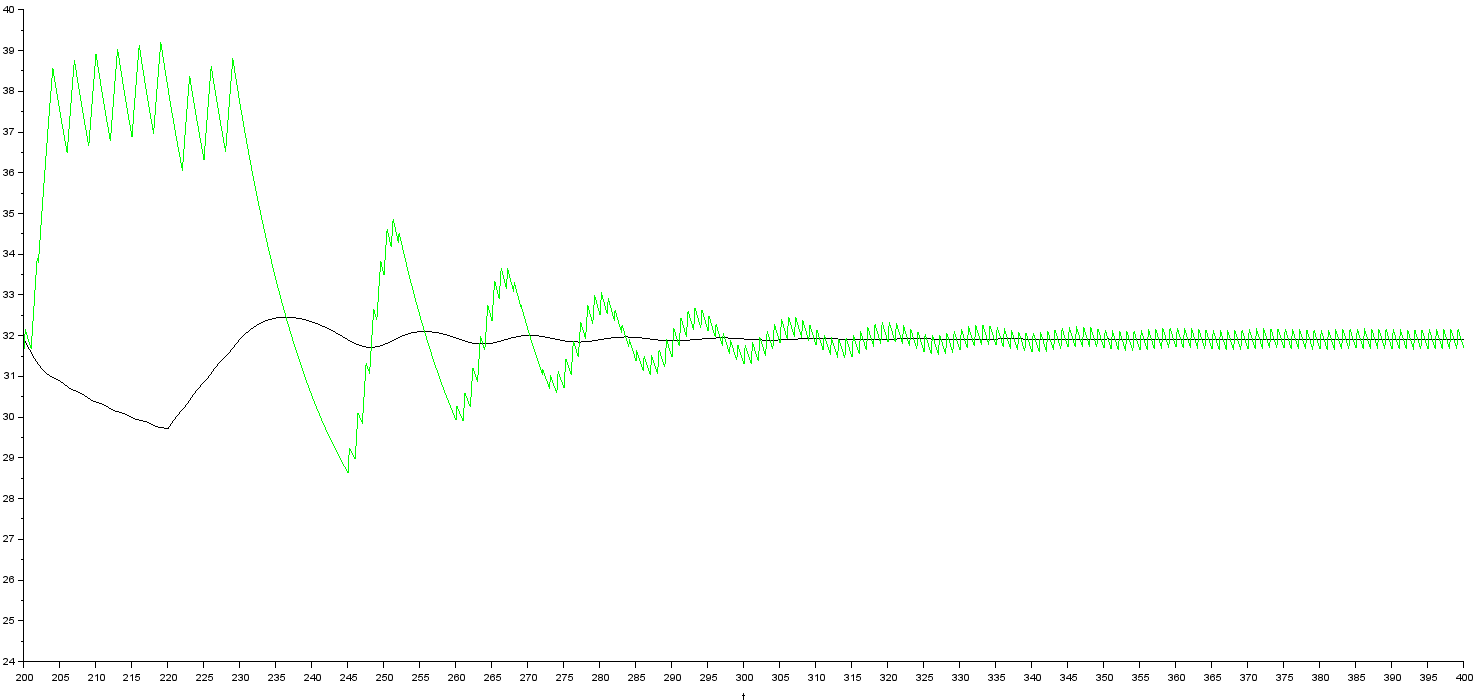


Figure 6 - Simulation of the entire system with Kp=2

Finally, we lowered the sampling period of the ADC for both thermistors and got slightly less oscillation in the system. However due to this change having an effect on the life time of the relays, we decided to not implement it as the net yield in oscillatory decrease is not worth the maintenance risks.

# **Design of the Low-End Controller**

## ***Arduino Internal LED***

(i-Buse, d-Halil, t-Berke, w-Servet)

A function named FSM\_flash() was written to create a blink-alive Led. This lights up the led inside the Arduino chip for 100 ms per every 2 seconds. When the LED flashes the letter ‘A’ is printed on the virtual Terminal.

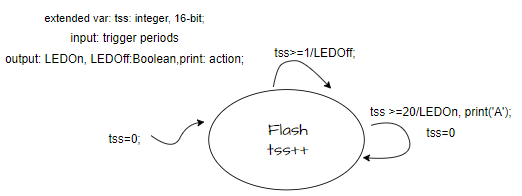


Figure 7 - FSM diagram for FSM\_flash ()

Implementation:

int tss;

void FSM\_flash (){

tss ++;

if(tss >=20) {

digitalWrite(13, 1);

tss =0;

Serial.print("A");

}

if(tss >= 1) {

digitalWrite(13 ,0);

}

}

The code has a variable named ‘tss’. We increment tss, and if it is higher than 20 (2 seconds) the LED is turned on and the tss is set back to 0. The letter ‘A’ is also printed here. If tss is greater than 1, meaning 0.1 seconds, we turn off the LED. This creates a blinking LED.

To test this function, we ran the simulation and we observed the LED as well as the terminal while keeping track of the time at the bottom of the page. Every 2 seconds the LED flashed and the letter ‘A’ was printed on the terminal so this functionality works correctly.



Figure 8- Test of the letter 'A' printing

In order to get a more accurate test than just observing the clock, we connected a virtual oscilloscope to the IO13 pin to see the LED flashing in a wave format. First, we need to set the oscilloscope for: - Horizontal sweep, Position 200, sweep time 200ms (=0.2s) per division; - Trigger Level 10, DC, rising edge, Auto, Cursors on. - Channel A, Position 120 (default) DC, sensitivity 2V per division.

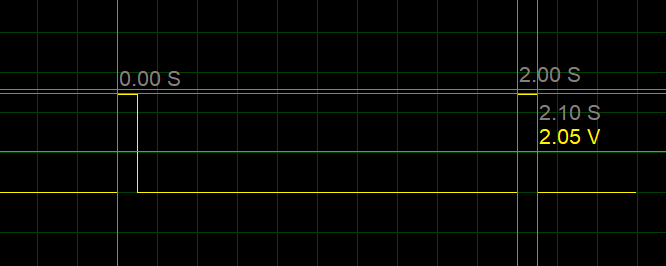


Figure 9 - Oscilloscope showing LED flashing

We can see clearly from Figure 9 that the LED flashes once every 2 seconds when we observe the time between the square waves. The time the square wave stays at 5 volts (ON state) is seen by the second measurement which is 0.10 seconds.

## ***Sensor Interfacing and ADC***

(i-Halil d-Berke t-Buse w-Abdulmalik)

In this section, we'll start by designing the actual hardware for Arduino-UNO, which will include the thermostats ThA and ThB to generate air temperature ("Ta") and infant temperature ("Tb"), respectively. To achieve this, we will add two 4700 Ohm NTC thermistors to Schematic Capture and connect them to the Arduino's AD0 and AD1 pins.

RT1 and RT2 represents the air temperature and infant skin temperature respectively. The connection of the pins of the Arduino, which we connect to read the values, is shown below. The calculations of ThA and ThB are contained in the function called ADC\_AB() which reads the thermostat values into two variables Ns0 and Ns1 which allow us to calculate ThA and ThB. Also, in the design we added 2 voltage probes to see the voltage of the NTC sensors.

After defining the t1m variable, we created an if state inside the loop to call the function every minute and wrote the ADC\_AB() function in this state. In addition, we need to limit ThA to 37 degrees Celsius so we added an if statement in this function.

void ADC\_AB (){

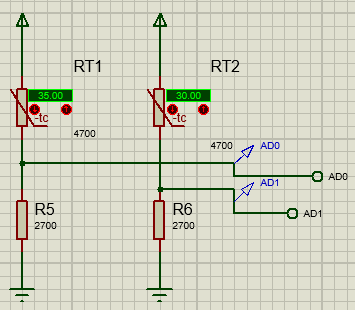


Figure 10 - Connectrion of the thermistors

Ns0 = analogRead(A0);

ThA = 10.\*T1 +Tcr+ (10.\*T2 -10.\* T1)\*(Ns0 -N1 \*1.)/(N2 -N1 \*1.);

Ns1 = analogRead(A1);

ThB = 10.\*T1 +Tcr +(10.\*T2 -10.\* T1)\*(Ns1 -N1 \*1.)/(N2 -N1 \*1.);

if (ThA >= 370){

ThA = 370;

}

}

By using SciLab Plot to display the sensors' response at 0, 5, 10, 45, and 50 °C temperatures, the performance of the sensors may be determined.

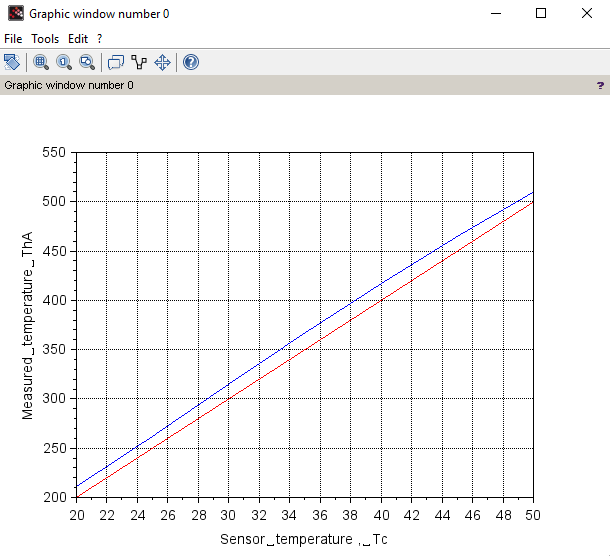


Figure 11 - Sensor Testing 1

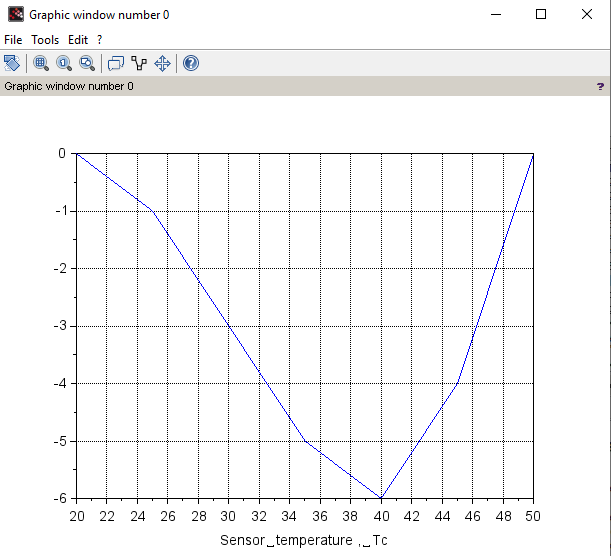


Figure 12 - Sensor Testing 2

The figures show the UART test results obtained using SciLab. As you can see, there is a shift of about 1.7 °C on the measured temperatures. Real-time data may not completely agree with theoretical information because of external factors. We should first fix the Ns0 and Ns1 measurements at 20 °C and 50 °C. To get the least variance around 35-37 °C, we must adjust ThA by adding 5 to the estimated number.

## ***UART Received Character Processing***

(i-Buse d-Halil t-Berke w-Berke)

All received characters are stored in a buffer by the interrupt-driven Arduino serial port. When you read a received character from the buffer, each character is delivered in fifo order. You receive -1 as the character value from an empty buffer.

To check the change in the displayed Kp and ThD values, we set the terminal to echo typed characters mode and sent the strings (4)p and (330)d. Additionally, it was checked to see if a space character would cancel the command and begin receiving the following one.

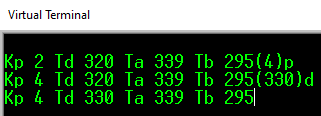


Figure 13 - Testing of the UART character processing

The FSM\_UART() function reads the input character by character. It sets N to zero upon receiving the character “(“. When the character is between 0 and 9, we calculate N with the formula 10\*N + C-48. After the closed bracket is received (“)”) either “d” or “p” will be entered which determines if we change the Kp value or the ThD value to the calculated N.

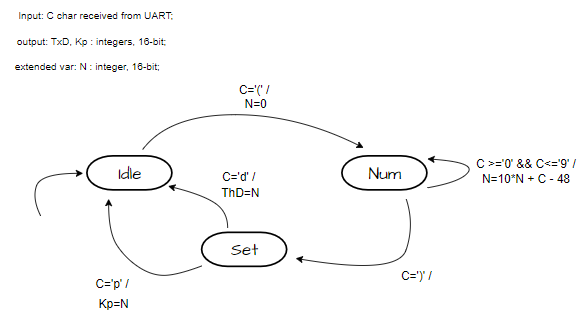


Figure 14 - FSM diagram for FSM UART()

void FSM\_UART (){

// FSM for received character actions

C=Serial.read ();

while(C!= -1){

if(Su==0 ) { // idle

if(C==40) {

N=0; Su=1;}

}else if(Su==1) { // num

if(C>47 && C<58) {N=10\*N+C -48;}

else if(C==41){ Su=2;}

else {Su=0;} // unexpected char

}else if(Su==2) {

if(C==100) {ThD=N; Su=0;}//"d"

else if(C==112) {Kp=N; Su=0;} //"p"

else {Su=0;}

}

C=Serial.read (); } }

## ***Control Task and PWM Operation***

(i-Berke d-Halil t-Buse w-Abdulmalik)

We have a function called PControl() which calculates the temperature difference between baby and desired temperature (e = ThD – ThB) and uses the percent-power formula (PP = Kp \*e) to get the PP value.

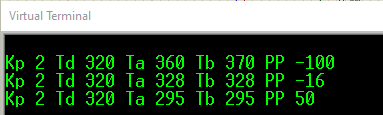


Figure 15 - Testing of the PP calculation

void PControl (){

// PP is calculated at every minute.

PP = Kp \* (ThD - ThB );

Serial.print(" PP ");

Serial.print(PP);

Serial.print(" ");

}

metin, yazı tipi, diyagram, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 16 - FSM FOR PControl ()

PP can be any positive or negative number, as you may have noticed. It is not upper- or lower-bounded. In the PControl() function, we added the following lines of code to keep the PP values between 1 and 99.

if(PP >99) PP =99;

if(PP <1) PP =1;

PP shall trigger the PWM for heater. We need to connect the heater relay to IO12 and place there a 220 Ω resistor and a LED as heater indicator. To control the heater status we created 2 functions "FSMpwm()" and "FSMtp2()". "FSMpwm()" is a Finite State Machine (FSM) for pulse width modulation (PWM) control. It increments "tpwm" and checks "ThA". If "ThA" > 369, "Htr" is set to 0 (heater off). If "ThA" <= 369, it enters a block. The heater shall be set to ‘1’ as long as tpwm counter is less than PP. If "tpwm" > 99, "Htr" is set to 1, because in the next run it will surely be less than PP, "tpwm" is reset to 0. If "tpwm" == "PP", "Htr" is set to 0 because we have reached the end of our duty-on time. Else when “ThA” is above or equal to 370, if "tpwm" > 99, it's reset to 0. If "PP" <= 1, "Htr" is set to 0 because this means that the baby is hotter than the desired temperature due to the calculation of PP. "Htr" value is written to digital pin 12 (LED control). If "tpwm" % 5 == 0, "tpwm" is printed to serial with "+" if "Htr" is true, or "-" if false. "FSMtp2()" increments "tp2" and calls "FSMpwm()" when "tp2" reaches 2, controlling the PWM timing.

**metin, diyagram, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**void FSMpwm (){

Figure 17 - FSM FOR FSMpwm ()

tpwm ++;

if(ThA > 369) Htr = 0;

if (ThA <= 369){

if(tpwm > 99) {Htr =1; tpwm =0;}

else if(tpwm == PP) {Htr =0;}

}

if (tpwm > 99) {tpwm =0;}

if (PP <= 1) { Htr = 0;}

// Output to LED

digitalWrite (12, Htr );

if(tpwm %5==0) {

Serial.print( tpwm );

if(Htr) Serial.print("+");

else Serial.print("-");

}

}

void FSMtp2 (){

// FSM for tp2tick output at every 0.1s

tp2++;

if(tp2 >=2) { tp2 =0; tpwm ++;

FSMpwm ();} // tp2tick calls FSMpwm ()

}

When we run the code, we tried all the possible conditions. We made ThA above 37 where the heater didn’t turn on. If the tpwm was lower than pp it did light. This meant that the more far away the baby is from the desired temperature in terms of being cooler, the heater turned on for a longer period of time. And finally, when the baby was above the desired temperature, the heater turned off because PP became 1.

## ***Design of the LCD***

(i-Buse d-Berke t-Halil w-Hazal)

In the design of LCD (Liquid Crystal Display) part and connected the IO pins 2-7 to it as well as a ground, a resistor and power. We used the 16x2 LCD to display Air temperature(A), Baby temperature(B), Desired temperature(D), Proportional gain(KP), Alarm status(AA), Power percent(PP) and Heater status(H). The value read(A) as 350 on the LCD screen corresponds to the actual temperature 35°C (same for other values ThB and ThD). If infant temperature (ThB) becomes higher than 37°C Heater(H) will become 0 thus heater will be Off. If the baby's temperature drops below 37°C and H will become 1 which means turning on the heater.

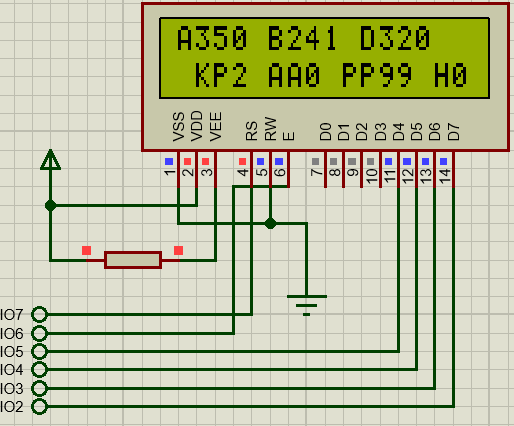


Figure 18 - The LCD panel

In Ardunio code, we first include the <LiquidCrystal.h> library in declaration part. Then we added a function called LCDdisp() to display the values on the LCD. In our first attempt, we realized that we could not print to the screen in a single line with the lcd.print feature, and we created the function by writing each variable separately:

void LCDdisp (){

lcd.clear();

lcd.print("A");

lcd.print(ThA);

lcd.print(" B");

lcd.print(ThB);

lcd.print(" D");

lcd.print(ThD);

lcd.setCursor(0 ,1);

lcd.print(" KP");

lcd.print(Kp);

lcd.print(" PP");

lcd.print(PP);

lcd.print(" H");

lcd.print(Htr);

}

## ***Design of the Alarm, UART Data Transmission and Testing***

(i-Halil d-Buse t-Berke w-Servet)

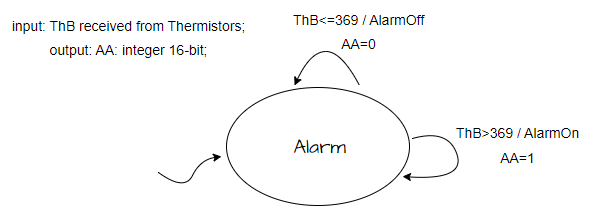
We implemented an alarm function that shows alarm status on the LCD screen. This function outputs "1" when the Baby surface temperature (ThB) goes above 37 degrees Celsius. If not, the alarm state stays as "0".

Figure 19 - FSM diagram for the alarm function

We added two more lines of code to the LCDdisp() function to implement this FSM:

lcd.print(" AA");

lcd.print(ThB >369);

The UART output was already configured to transmit the gain Kp and the temperatures ThA, ThB and ThD during previous phases. It also transmits power in percentage. Only the cover status and alarm condition are missing from the data string. After declaring the SwCov variable we connected a switch at IO10 and set it to pinMode(11, INPUT\_PULLUP). When we open or close the switch it is going to be 1 or 0 respectively. And we modified the name of UART\_PrintT() function ,which works for printing some values to the virtual terminal, to UARTprintAll().

void UART\_printAll (){

Serial.println ();

Serial.print("Kp ");

Serial.print(Kp);

Serial.print(" Td ");

Serial.print(ThD);

Serial.print(" Ta ");

Serial.print(ThA);

Serial.print(" Tb ");

Serial.print(ThB);

Serial.print(" PP ");

Serial.print( PP );

Serial.print(" AA ");

Serial.print( ThB >369 );

SwCov= digitalRead(11);

Serial.print(" C ");

Serial.print(SwCov);

}

The loop section was modified to accommodate the following code which allows for the change of the cover status without having to wait for the regular 60 second period which is for all other data:

oldSwCov=SwCov;

SwCov= digitalRead (11);

if(SwCov != oldSwCov ) {

Serial.print(" C ");

Serial.print(SwCov );}

When we run the code it shows that our codes and design are correct. When you change the status of switch, it first shows it as extra cover status and for the other reading line it shows the new cover status.

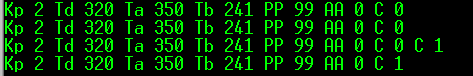


Figure 20 - Testing of UART transmission of alarm and cover status

Overall testing of the Low-end controller was also completed. We ran the entire code at once and checked to see if all the functionalities are working. We checked all possible conditions for the temperatures and for the cover switch. All the boxed were ticked so it was time to move on to the next part.

## ***Code for Setup and Loop***

(i-Buse, d-Berke, t-Halil, w-Abdulmalik)

void setup () {

peripheral\_setup();

lcd.begin (16, 2);

Serial.begin (9600);

pinMode (13, OUTPUT );

pinMode (12, OUTPUT );

pinMode(11,INPUT\_PULLUP);

pinMode(10,OUTPUT);

lcd.print("hello ");

tss =0;

analogReference(DEFAULT );

T1=20; N1 =333; T2=50; N2 =637; Tcr=0;

Su=0; // FSM -UART state

Kp=2; ThD =320;

}

void loop() {

peripheral\_loop();

delay(100);

FSM\_flash ();

t1m ++;

if(t1m >= 600) { // tasks at every minute

t1m =0;

ADC\_AB ();

FSMtp2();

PControl();

UART\_printAll();

LCDdisp ();

}

FSM\_UART();

FSMpwm();

oldSwCov=SwCov;

SwCov= digitalRead (11);

if(SwCov != oldSwCov ) {

Serial.print(" C ");

Serial.print(SwCov);}

}

# **Node-RED Implementation**

## ***Configuring Proteus***

(i-Aydın, d-Cahit, t-Halil, w-Abdulmalik)

We select the desired COMPIM ISIS module and configure it to use COM10. Additionally, we adjusted the baud rates of COM10 serial ports to ensure optimal communication between Proteus and Node-RED. We also set the data bits to “8” and the parity to “NONE”. Then we connected the IO0 AND IO1 pins to the RXD and TXT pins of our COMPIM module. Then we ran the code. The code must be running throughout this process. The light lit up meaning the module is working.

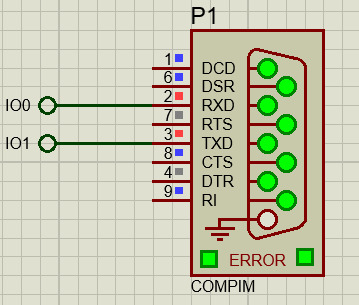


Figure 21 - COMPIM

## ***Configuring VSPE***

(i-Cahit, d-Aydın, t-Buse, w-Berke)

To establish a connection between Proteus and Node-RED, we employ VSPE as a virtual serial port emulator. VSPE allows us to create virtual serial ports, which act as software-based representations of physical ports, enabling data transmission between the two applications. Firstly, we install and configure VSPE on our system. This involves downloading the latest version of VSPE from the official website and following the installation instructions provided [2]. Once installed, we configure VSPE to create two virtual serial ports, namely COM10 and COM11. Then we use the serial redirector tool to direct the COM10 port coming out of our proteus to the COM11 port that will be used for the Node-RED.

## ***Node-RED software***

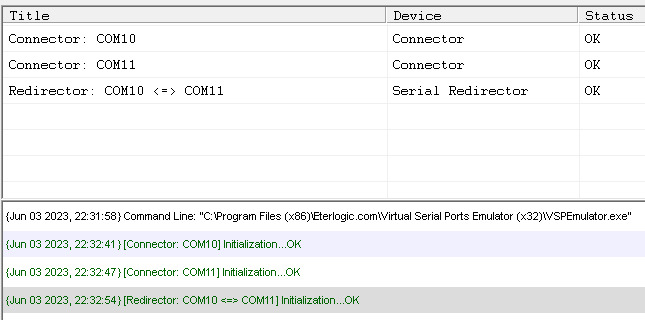


Figure 22 - VSPE UI

(i-Halil, d-Cahit, t-Berke w-Aydın)

Each and every team member had Node-RED components downloaded and installed on their computers so everything was ready. We opened the command prompt in Windows operating system and typed in the command “node-red”. We copied the URL address that was shown after the command processes.

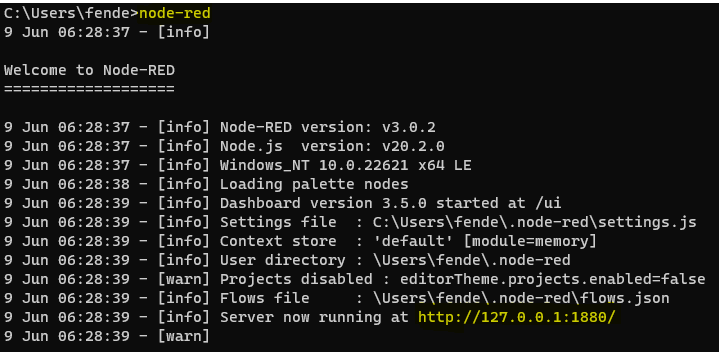


Figure 23 - Node-RED CMD interface

We then pasted this address into our browser and got access to the dashboard. Here we used serial input, serial output, split, gauge dashboard, chart dashboard and slider nodes (also debug node used but it is not necessary to add, it used for the checking the values). The data coming in is the data shown on the virtual terminal to we used some strings to split this information into parts we can use to display in a user-friendly manner. So, we used split nodes and connected them to the serial input as well as a debug node to test if the data is flowing in. Then we added the strings into the split nodes which we use to split the data into blocks. Moreover, we placed some charts, gauges and sliders to show this information. In Table 1, the strings we used to split the data is available. In figure 23 the connections are available.

Table 1 - String table to split function

|  |  |
| --- | --- |
| **Data** | **Meaning** |
| Kp | The value of proportional control |
| Td | Desired Temperature |
| Ta | Air Jacket Temperature |
| Tb | Baby Temperature |
| PP | Percent-Power |
| AA | Alarm status for air jacket temperature |
| C | Cover Status |

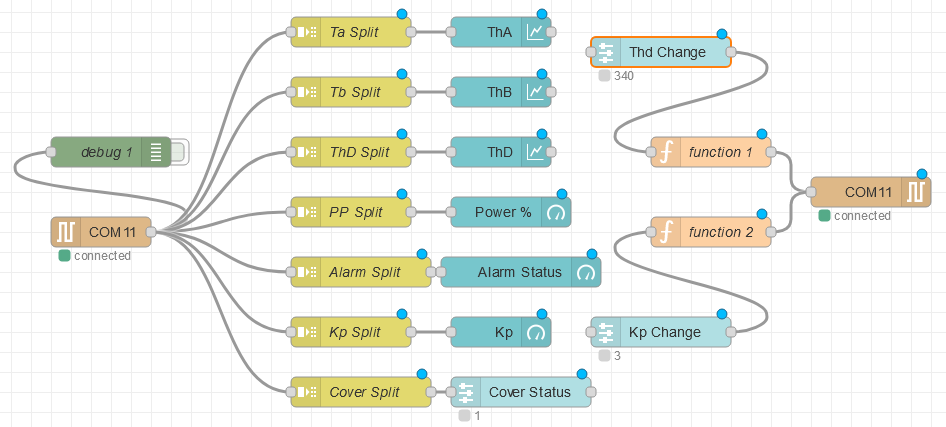


Figure 24 - Flow diagram on Node-RED

We entered the same URL we used before but with “ui” after the dash to have access to the user interface. Here we saw that our flow diagram was correct and we were able to extract and show every piece of data that we wanted to show as is clear from Figure 24.

We can see that the overall system works well. Every change we make from proteus is inputted into the Node-RED software and the information is shown in a user-friendly manner. All our low-end parts work fine as when we did an overall test of the system with every condition possible, the system reacted in the way we expected it to with the correct outputs.

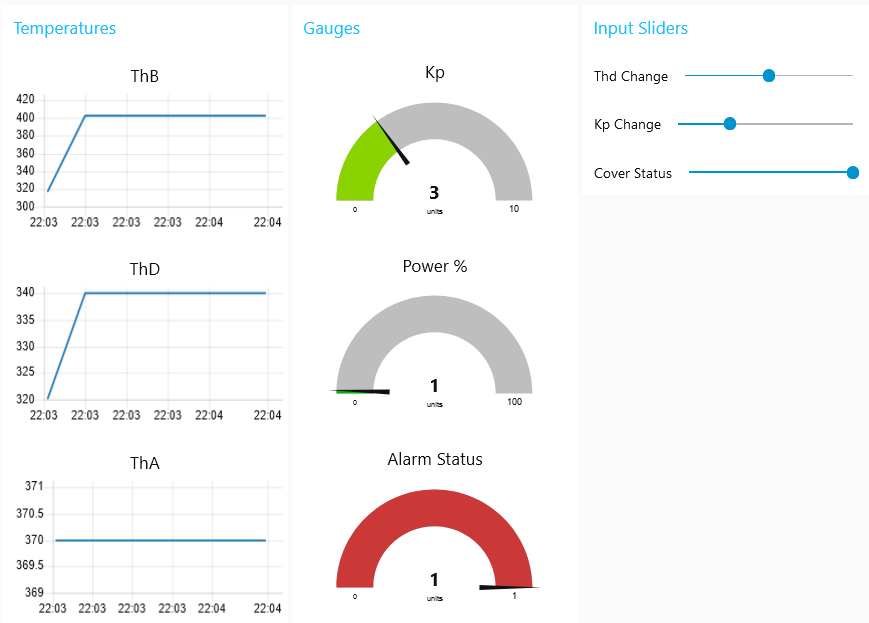


Figure 25 - UI of the Node-RED software

# **Conclusion**

**5.1 Berke Emerce**

In this project, I was the idea owner and documenter for the overall system design, documenter for the physical SciLab simulation, tester for the Arduino LED flash function, designer for the sensor interfacing and ADC, tester for the UART character processing, idea owner for the control task and PWM operation, designer for the LCD, tester for the alarm function and cover switch, designer for the "setup" and "loop" parts of the code and tester for the Node-RED software. Additionally, I managed the team by conducting pre-project meetings to assess everyone, assigning tasks based on my assessment, managing the tasks I assigned everyone and giving feedback, helping with the given tasks and by conducting regular meetings to see how the project is progressing and finishing any unfinished parts of the project. In the parts I undertook, the PWM operation and control task was the most challenging where I spent days to figure it out however in the end it all works correctly. I believe that the entire project was a success overall. All aspects of the project work as intended and the team spirit was kept intact throughout the process. I think that this project was a very helpful one in terms of aiding with understanding the course itself. I also gained a lot of experience as a leader which will help me in the future.

**5.2 Abdulmalik Olaoye Raji**

In conclusion, my active involvement in defining design objectives, conducting thorough testing of physical and cyber-physical systems, and documenting key aspects like Sensor Interfacing, ADC work, Control Task, and PWM Operation contributed to the overall success of the project. Despite encountering minor challenges during cyber-physical system testing, with the help of my teammates, we overcame them effectively. I also learned how to document and format an engineering report properly. The project as a whole achieved remarkable success, showcasing our collaborative efforts and meticulous attention to detail. I am grateful for the experience gained and proud of our achievements as a team.

**5.3 Hazal Aydın**

Throughout this project I was the idea owner for the physical SciLab simualtion, Designer of the whole SciLab simulation with the cyber parts added, researched and wrote the social impacts of the IncuBaby as well as wrote the process of LCD design. My parts were a success and work correctly. Our project had some issues at first version but after working on our mistakes according to the feedback we received from the team leader, we were able to achieve an overall successful project. Overall, this project enhanced my understanding of embedded systems.

**5.4 Halil Yüksel**

In this project, I undertook the idea owner of design objectives, introduction, sensor interfacing and ADC, design of the alarm, UART data transmission and testing Node-RED software parts and the designer of Arduino internal LED, UART received character processing, control task and PWM operation parts and the tester of overall system, design of the LCD, Node-RED implementation and the loop and setup parts of our code. These parts were successfully completed. Finally, we worked as a team and made a working embedded system for Baby Incubator. As a result, I believe we have done something to make people's lives easier.

**5.5 Buse Alasköz**

In the end, working as a team provided a unique experience. Collaborating with others, in addition to individual work, brought together multiple perspectives and led to the generation of more creative ideas. Throughout the project, I took part in the design of the overall system and alarm (Data Transmission), having an idea on the Arduino internal led, UART received character and LCD parts as well as the “loop” code, and finally in some parts of testing and document preparation. While we weren't able to come up with a perfect design, I was largely satisfied with the system's success. Despite encountering initial challenges, we successfully completed the project. By acquiring knowledge on utilizing significant high/low-level designing tools, we gained valuable insights that proved to be highly beneficial by the project's conclusion.

**5.6 Cahit Osmanlı**

In summary, our project is very successful overall. We used a variety of tools, collaborated as a team, and got the result we wanted. I contributed in a variety of areas, but I mainly concentrated on design resources and requirements, designing the COMPIM connection, having the idea for the VSPE configuration, and designed the node-RED software. By the help of my teammates everything is very successful. It was a pleasure working on this project, and I sincerely appreciate the contributions of my teammates and many thanks to our team leader, Berke

**5.7 Aydın Eğitmen**

In conlusion , I was the documenter for the SciLab simulation of the whole system, designing the VSPE configuration and documenter for the Node-RED software. Figuring out the connection between VSPE and COMPIM ISIS from our ardunio uno program and designing the serial ports of the VSPE was simple and easy. I believe that this project was incredibly useful to improve our teamwork. Our project works successfully without errors or problems in every part and every part was done diligently

**5.8 Servet Nursu Boztaş**

During my involvement in this project, I mostly worked on designing the cyber part on Scilab and I helped with my teammates modelling analysis simulation of the whole system. I also documented some parts of the project. Working on my parts was fulfilling because I believe I learned a lot about the subject. Generally, our project was a success and I gained valuable time-management lessons from it. I thank all my teammates and leader for this successful project

# **References**

[1] M. Bodur, CMPE320 Embedded Systems Design Technical Requirements for IB04 Incubator project, G-Magusa: M-Bodur, 2023.

[2]Eterlogic software: Downloads page. (n.d.). Eterlogic - Welcome to www.eterlogic.com website. <https://www.eterlogic.com/Downloads.html>