Incubaby (Ib03) Design of Simulation of LOW-End Controller for a Premature Infant Incubator Connected to NodeRed on a PC for High End User Interfacing Through Web Dashboard

EASTERN MEDITERRANEAN UNIVERSITY

CMPE320 EMBEDDED SYSTEM DESIGN

TEAM-3 FIRST REPORT Revision 1

Date of Submission: **24/06/2023**

TEAM LEADER:

3.1 İBRAHİM BERK ALAGEYİK, 20000125

TEAM MEMBERS:

3.2 EGE BARLAS, 20000051

3.3 MOHAMMAD IMAD KHAN, 20800922

3.4 AHMET FATİH SARUHAN, 20001788

3.5 EMRE ÇAKIR, 20001797

3.6 AHMED ALGHUSSEIN, 20800834

3.7 RAMAZAN KARATAŞ, 19000047

3.8 İSFENDİYAR BERKAN KASAPOĞLU, 18330810

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

OVERALL SYSTEM……………………………………………………………… 3

INTRODUCTION…………………………………………………………………. 4

Social Impacts ………………………………………………………………………4

DESIGN RESOURCES…………………………………………………………….5

Design Requirements………………………………………………………………. 6

MODELLING AND ANALYSIS FOR CONCEPTUAL DESIGN..………………8

Test A: Simulation of Physical System……………………………………………...8

Test B: Simulation of Embedded system (Cyber-Physical part) ………………….. 10

DESIGN OF DIGITERM…………………………………………………………..12

Starting an Arduino Project with Firmware...............................................................12

Test of Flashing LED Design Step.............................................................................13

Sensor Interfacing and ADC......................................................................................14

Test Readings................. …………………………………………………………...15

UART Received Character Processing....…………………………………………..16

Test of Received Character FSM……………………………………….......………17

Control Task...............………………………………………………………………17

Testing Controller and PWM Operation....................................................................19

Test of Heater off Action on ThA > 37 °C................……………………………….19

LCD Display ………………………………………………………………………..20

Firmware for LCD Display...............................................................………………..20

Testing Data Displayed on LCD,,…………………………………………………..21

UART Data Transmission................... ……………………………………….......... 22

Testing UART Transmit Data.........………………………………………………...23

NODE-RED IMPLEMENTATION…………………………………………….......24

CONCLUSION..........................................................................................................28

REFERENCES……………………………………………………………………...30

**DESIGN OBJECTIVES**

(i-İbrahim v-Alghussein e-Imad)

# The creation of a system for infant incubators that properly measures and controls the temperature inside the inner chamber is the main objective of this project. We use proportional control law to maintain the appropriate temperature. The system automatically ensures that ThA and ThB never exceed the temperature ThD, which is set at 32 oC. It keeps the baby safe by sounding the automated alarm anytime ThB rises above 37 oC. The terms "air temperature" and "baby temperature," respectively, are ThA and ThB. The Arduino Uno and NodeRED are also connected via a Virtual Serial Port Emulator (VSPE), which delivers 8-bit data at 9600 Baud rate with no parity. This makes it possible for the Arduino UNO firmware source code settings to be displayed on NodeRED debug windows. In addition, when temperatures are altered in the schematic, a matching display is visible on the LCD. The LCD display displays the target temperature ThD as well as the temperatures of the thermistors ThA and ThB.

# **OVERALL SYSTEM**

(i-Alghussein d-Ege e-Isfendiyar)

A picture containing diagram, plan, line, text

Description automatically generated

Figure 1: Overall view of the system

# The Arduino-Uno based system for controlling temperature levels in the infant incubator consists of several components. Firstly, temperature sensors to monitor the Air-Heater temperature (ThA) and the baby's temperature (ThB). Additionally, an LCD display is incorporated to present information, including the desired temperature, current temperature, heater on-off status, and current mode selection. The temperature control is achieved using a PWM switching heater that is connected to the Arduino-Uno board and is responsible for adjusting the temperature inside the incubator. A serial port is available to establish communication with Node-Red on a PC, where a dashboard is set up to display essential information for monitoring and controlling the incubator's temperature. Furthermore, there is a lighting LED that flashes in yellow color during the project's operation which indicates the heater status.

# **INTRODUCTION**

(i-Imad v-Ahmed e-Alghussein)

# This report is about our attempt to design and test a DigiTerm 03 Low-end temperature controller unit for an infant incubator. We have documented each step along the way that has led us to our final product. Each of us has made an effort in understanding the workings of our product by implementing prototyping practice. Testing has been the hallmark of our project to maintain the healthy functioning of product.

# **Social Impacts**

(i-İsfendiyar v- Imad e-Ibrahim)

In order to maintain the body temperature of babies and safeguard their health, baby incubators are utilized to provide a controlled and safe atmosphere. The benefits this project has to offer in terms of ensuring infants are provided optimal temperature and a maintainable suitable environment around them. These systems are designed with features such as alarms, LED lights and screens that display the baby’s body temperature and the temperature of the air heater circulating within the chamber.

These systems offer high level of safety and care for infants compared to relying on human intervention. These systems help save time, effort and money while improving the overall well-being of the babies in their crucial early stages of life.

# **DESIGN RESOURCES**

(i-İsfendiyar v-Ibrahim e-Imad)

The IncuBaby project was built using several essential applications that helped us in the implementation and testing processes. These applications are as follows:

* Our laptops/desktop computers were used to run the various software that we needed to design this project.
* 7 team members and a project leader. Total 8 members.
* We used SciLab, it is a free, open-source numerical computational package and programming language. We used it for the purpose of modelling and conceptual thereby creating a model of the physical and cyber systems using the SciLab XCOS environment.
* We used Proteus Professional 8, it is a comprehensive software used for drawing schematics, PCB layout, coding and simulating schematics. We used it to design and simulate the Arduino-UNO module, thereby completing the low-end embedded control unit aspect of the project.
* Terminal: A virtual terminal included in Proteus software is used for debugging purposes.
* Oscilloscope: We used an oscilloscope to keep track of time during our simulations in Proteus.
* We used VSPE, it is tool used for the creation, debugging and testing of applications that use serial ports. We used VSPE to create virtual devices such as COM10 and COM11. So that Node RED and Arduino were virtually connected using the VSPE software.
* We used Node-RED, it is a flow-based development tool primarily used for visual programming, particularly in the context of the Internet of Things (IoT). We used it to construct a dashboard that displays system components based on specified time intervals, facilitating notetaking and observation.

By making use of these applications, we were able to effectively implement and test the IncuBaby project, thus creating a fully functional and reliable infant incubator system.

# **DESIGN REQUIREMENTS**

(w- Ibrahim)

Based on the design specifications given in previous laboratories (Lab 5-6-7) and technical requirements file shared [1], the following requirements are identified:

1. Cyber-Physical Modelling:[2]

* Develop an overall model for an incubator that maintains body temperature (ThB) within the desired temperature (ThD) range. The control law should ensure that ThB remains stable without overshooting above 37 degrees Celsius.
* Consider using a proportional (P) control approach, where the desired temperatures are set to compensate for steady-state offsets.
* The model should include the temperature dynamics of the air-jacket and the incubation chamber, with sensor signals for ThA (air temperature) and ThB (body temperature).
* Incorporate control over the percentage extent of the air-heater in the air-jacket.
* Implement an inhibiting action for the ThB sensor when ThB exceeds 37 degrees Celsius.
* Test and verify the model with the following criteria:
* Ensure that the model dynamics align with the specified time constants provided in the project documentation for each component.
* Validate that the control action successfully maintains the body temperature at the desired level, even in the presence of a malfunctioning temperature sensor that indicates very cold temperatures. Success is defined as ensuring that the body temperature never exceeds 37 degrees Celsius.

2. Low-End Controller Unit Design:[3]

(w-Ege)

* The system is developed on Arduino-Uno R3 board with 16 MHz clock.
* It should have a flashing LED (internal LED of Arduino) that flashes for 20 ms in every 0.1 seconds.
* It should have two 4k7 NTC thermistors with proper linearization resistors to read temperatures between 20-45 °C with 0.5 °C precision.
* UART should receive ASCII commands from user in terminal.
* There should be a proportional feedback gain Kp for control action. The following task should be done:
  + Measure ThA and ThB temperatures. Temperatures can be set manually by setting the thermistor’s resistance to corresponding value.
  + Calculate e = ThD – ThB
  + Calculate PWM duty-time Td = 20\*PP / 100 seconds and keep the heater on for Td seconds.
* Display the following values on a 16x2 LCD display:
  + ThA, ThB and ThD temperatures: int 10 \* temperature in °C
  + Proportional gain Kp integer
  + Alarm status AA according to ThB < 37 °C
* UART should transmit:
  + Temperatures ThA, ThB, ThD
  + Proportional gain Kp
  + Alarm status AA

3. Node-Red Design on PC:[4]

* Node-Red should provide a 9600 Baud, 8 data, no parity, one stop serial port.
* It should retrieve ThA, ThB, PP values and alarm status from Arduino through serial port connection.
* It should transmit ThD value (32, 34 or 36 °C).
* It should display ThA, ThB, ThD, Kp and PP values as well as cover and alarm status on the interface.
* Users should be able to adjust values of ThD and Kp on the interface.

# **MODELLING AND ANALYSIS FOR CONCEPTUAL DESIGN,**

(i-Imad e-Alghussein)

We start with the simulation of the cyber physical system as it is a vital part of an embedded system. We have used the SciLab’s XCOS design environment to run the following tests and simulations.

**Test A: Simulation of Physical System**

(i-Imad d-Ibrahim t-Isfendiyar e-Alghussein)

The signal flow diagram was adjusted according to the provided settings:

* A gain block was introduced to set the output temperature of the air heater to 40°C.
* A step function was applied to set the environment temperature to 25°C.
* The temperatures from the air heater and environment were combined using a summation block.
* The air heater's output temperature was modified using a CLR (Continuous Linear Reset) block, which operates based on the formula (1/(1 + 12 \* s)).
* A GenSQR block was implemented to generate a signal of -10°C every 200 minutes, simulating the action of opening the cover.
* The output of the air heater and the GenSQR signal were summed together using another summation block.
* A CLR block was employed, utilizing the formula (1/(1 + 30 \* s)).

These adjustments reflect the specified modifications made to the signal flow diagram based on the provided settings.

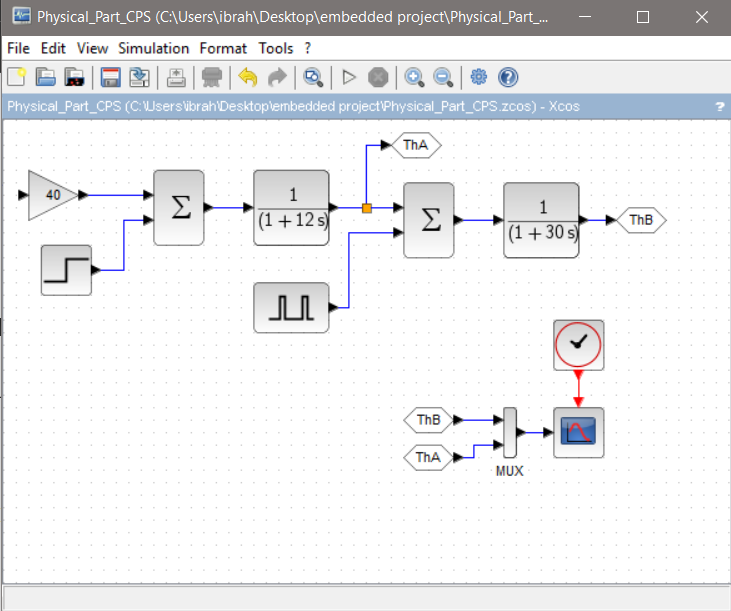


Figure 2: Signal Flow Diagram of the Physical System

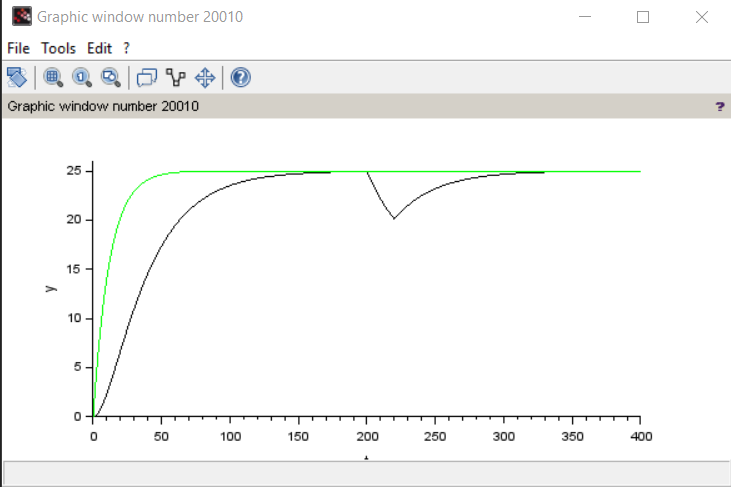


Figure 3: Test A Simulation Result

Result explanation:

According to Figure 3, the incubator lid opens for five minutes at a time equal to 200 minutes before closing once more.

**Test B: Simulation of Embedded System (Cyber-Physical Part)**

(i-Alghussein d-Ibrahim t-Imad e-Ege)

The Cyber part of the design incorporates the following settings:

* A PID control law combined with Pulse Width Modulation (PWM) is utilized to regulate ThA (air temperature) and maintain it within the bound of 37°C.
* Sample Hold (S/H) blocks are employed to convert the analog signals ThA and ThB from their continuous form to digital representation.
* A Sawtooth function is employed to compare the temperature value of ThA with 37°C. Its output is a logical signal that ensures ThA never exceeds 37°C.
* A UBound block is utilized to impose an upper limit of 37°C on ThA, restricting its temperature to that value.

A picture containing text, diagram, screenshot, plan

Description automatically generatedThese settings are implemented in the Cyber part of the design to control and maintain the air temperature (ThA) within the specified bounds of 37°C.

Figure 4: Signal Flow Diagram of the Cyber Part of the System

A picture containing text, diagram, font, line

Description automatically generated

Figure 5: Test B result simulation for Kp = 20

A green and black graph

Description automatically generated with low confidence

Figure 6: Test B result simulation for Kp = 2

Figure 6 above shows that compared to Kp = 20 (Figure 5), Kp = 2 results in a more accurate system since the errors decrease with time as the system approaches steady state (ThD=32°C). That is, it gradually moves closer to the steady state, as opposed to Kp=20, where the mistakes do not diminish over time. Because of this, a promotional control Kp = 2 is better suited for the system. Also, although low the sampling periods may give more accurate results, lowering it too much may damage the relay, so we kept the periods at 1 with 0 installation time.

**DESIGN OF DIGITERM**

**Starting an Arduino Project with Firmware**

(i-Ramazan d-Ege t-Emre e-Ege)

For implementing the simulation of the project, we created an Arduino project named “Project\_Part1” for Arduino-Uno and Ard.AVR (Proteus) compiler. The clock rate of Arduino UNO was set to 16 MHz by default. For test purposes we started 9600 baud serial port through UART. Then for flashing internal LED we draw the FSM chart in Figure 7 and to implement it wrote the code below:

A picture containing text, diagram, font, line

Description automatically generated

Figure 7: FSM diagram for flashing internal LED

// ----------declarations -----------

int tss;

void FSM\_flash (){

// FSM for flashing light.

tss ++;

if(tss >=20) {

digitalWrite (13, 1);

tss =0;

Serial.print("A"); // Test of transmit "A"

}

if(tss >= 2) { digitalWrite (13 ,0); }

}

// ----------setup -------------------

Serial.begin (9600);

pinMode (13, OUTPUT );

tss =0;

// ----------loop --------------------

delay (100);

FSM\_flash ();

**Test of Flashing LED Design Step**

(t-Emre)

To test the flashing LED and character transmission, we connected IO1 to a virtual terminal and IO13 to a virtual oscilloscope with following settings:

* Horizontal; Sweep, Position: 200, Sweep time: 200 ms – Trigger; Level: 10, DC, Rising edge, Auto, Cursors on – Channel A; Position: 120, DC, Sensitivity: 2V

A screenshot of a video game

Description automatically generated with medium confidence

Figure 8: Virtual terminal and oscilloscope on schematic capture

A picture containing screenshot, line

Description automatically generated

Figure 9: Oscilloscope readings

**A screen shot of a computer

Description automatically generated with medium confidence**

Figure 10: Terminal output

We observed the outputs at Figure 9 & 10 as well the internal LED flashing 200 milliseconds in 2 seconds intervals. At this point, we saved our file as “Project\_Part2” to move onto the next part.

**Sensor Interfacing and ADC**

(i-Emre d-Fatih t-Ege v-Ramazan)

In this part we used two thermistors ThA and ThB to get air temperature and baby skin temperature, respectively. We placed two 4700 Ohm NTC thermistors and two 2700 Ohm resistors connected to AD0 and AD1 of Arduino. Thermistor ThA will give the air temperature and ThB will give baby skin temperature. The design of the thermistors is shown in Figure 11. We also keep ThA from exceeding 370. The codes for calculating the values of ThA and ThB and for transmitting the readings are as follows:

// ----------declarations -----------

int Ns0 , Ns1 , ThA , ThB , t1m;

int T1 , T2 , N1 , N2 , Tcr;

void ADC\_AB (){

// Default ADC resolution is 10-bit

// Read ADC , and Calculate Temperature

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;

}

void UART\_printT (){

Serial.println ();

Serial.print("N0");

Serial.print(Ns0);

Serial.print(" Ta ");

Serial.print(ThA);

Serial.print(" N1 ");

Serial.print(Ns1);

Serial.print(" Tb ");

Serial.print(ThB);

}

A picture containing text, diagram, line, plot

Description automatically generated

Figure 11: Schematic capture of ThA and ThB thermistors

**Test Readings**

(t-Ege)

To test the performance of the sensors we plot their response at 0-50 °C temperatures using SciLab Plot. Figure 12 shows that there is a slight difference between measured temperatures and sensor temperatures. To reduce the deviation to minimum around 35-37 °C, we change the Tcr value from 0 to 5 and obtain the graph given in Figure 13.

A picture containing line, plot, text, slope

Description automatically generated

Figure 12: Measured Temperature vs. Sensor Temperature

A picture containing line, plot, diagram

Description automatically generated

Figure 13: Deviation vs. Sensor Temperature

At this point, we saved our file as “Project\_Part3” to move onto the next part.

**UART Received Character Processing**

(i-Fatih d-Ege t-Emre e-Fatih)

For UART received character processing we took the FSM chart given in Figure 14 as a reference from Lab7 document to implement it in Arduino Sketch Code with additions to the existing code.

A picture containing text, diagram, circle, font

Description automatically generated

Figure 14: FSM diagram for UART received character processing

// ----------declarations -----------

int C, Su , N, Kp , ThD;

// ----------setup -------------------

Su=0; // FSM -UART state

Kp=10; ThD =320;

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 (); }

}

// ----------loop --------------------

delay (100);

FSM\_flash ();

t1m ++;

if(t1m >= 60) { // tasks at every minute

t1m =0;

ADC\_AB ();

UART\_printT ();

}

FSM\_UART ();

**Test of Received Character FSM**

(t-Emre)

We started the simulation and enabled echo typed characters. We observed that the values of Kp and ThD changed by respectively sending “(2)p” and “(330)d” strings to the terminal as seen in Figure 15.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 15: Output for UART received characters

At this point, we saved our file as “Project\_Part4” to move onto the next part.

**Control Task**

(i-Emre d-Fatih t-Ege e-Ege)

To calculate percent power PP we first calculate the error by e = ThD – ThB and then multiply it with Kp in the following function.

// ----------declarations -----------

int PP, tp2, tpwm, Htr, tp6

// ----------setup-------------------

pinMode(12, OUTPUT);

tpwm =0; tp6 =0; Htr =0;

void PControl (){

// PP is calculated at every minute.

PP = Kp \* (ThD - ThB );

if(PP >99) PP =99; if(PP <1) PP =1;

Serial.print(" PP ");

Serial.print(PP);

Serial.print(" ");

}

PControl function generates PWM duty time periodically. To control the working time of heater we create the FSMpwm() function. We have the variable tpwm to keep track of time. When tpwm < PP, the heater will be on.

A picture containing text, diagram, font

Description automatically generated

Figure 16: FSM diagram for FSMpwm function

void 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 <= 0) { Htr = 0;}

// Output to LED

digitalWrite (12, Htr );

if(tpwm %20==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 ()

}

**Testing Controller and PWM Operation.**

(t-Fatih)

We used an LED to display the status of the heater. Also, tpwm values and status of the heater will be displayed on the virtual terminal periodically as well. ‘+’ on the terminal means the heater is on, ‘-‘ means it is off. If ThA is greater than 369, Htr (and LED) will stay off regardless of current values of tpwm and PP. We started the simulation and set the Kp value to 2 by sending the string “(2)p” to the terminal. Figure 17 shows the output we observed.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 17: Virtual terminal output and Htr status on LED

**Test of Heater off Action on ThA > 37 °C**

(t-Fatih)

In the code there is also an if statement to keep the heater off once the air temperature exceeds 37 °C. In that case, the heater and the external LED light will always stay off regardless of the PP value.

At this point, we saved our file as “Project\_Part5” to move onto the next part.

**LCD Display**

(i-Ege d-Fatih t -Ramazan e-Ege)

Since we were going to print the heater status on the LCD, we first commented out the testing lines of PWM. Then we used LM016L from components and connected the respective terminals to respective pins of LCD as seen in Figure 18.

**A picture containing text, diagram, rectangle, line

Description automatically generated**

Figure 18: LCD Display on schematic capture

**Firmware for LCD Display**

(i-Ege d-Fatih t-Ege e-Fatih)

We declared LiquidCrystal.h library and initialized the LCD in 16x2 character mode by adding the codes below:

// ----------declarations -----------

#include <LiquidCrystal.h>

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

// ----------setup -------------------

lcd.begin (16, 2);

.

.

.

lcd.print("hello");

Upon test simulation, we received the “hello” message displayed on LCD screen as in Figure 19.

A picture containing text, screenshot, font, display

Description automatically generated

Figure 19: Test simulation result

After testing, we created the following function to display our actual information on the LCD:

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("AA");

lcd.print(ThB >369);

lcd.print("P%");

lcd.print(PP);

}

We put this function in the loop so that the LCD display is updated once every minute.

**Testing Data Displayed on LCD**

(t-Emre)

We ran the simulation again and observed the output on LCD display in Figure 20.

A screenshot of a computer

Description automatically generated

Figure 20: Values displayed on terminal and on LCD display

At this point, we saved our file as “Project\_Part6” to move onto the next part.

**UART Data Transmission**

(i-Fatih d-Fatih t-Ege e-Ege)

In this part we completed the missing part on the data string, which was the cover status. An SPST switch is placed on schematic capture with IO11 and IO10 connected to each side of the switch.

For configuration and reading the cover status as well as its change following codes are added:

// ----------declarations-------------

int SwCov, oldSwCov;

// ----------setup -------------------

pinMode(11,INPUT\_PULLUP);

pinMode(10,OUTPUT);

void UART\_printAll(){

...

SwCov= digitalRead (11);

Serial.print(" C ");

Serial.print(SwCov );}

// ----------loop--------------------

...

oldSwCov=SwCov;

SwCov=digitalRead (11);

if(SwCov != oldSwCov) {

Serial.print(" C ");

Serial.print(SwCov );}

We also commented out the printing codes in PControl and put them in UART\_printT(), renaming it to UART\_printAll() to have all remaining information to be printed in one function.

**Testing UART Transmit Data**

(t-Ege)

Figures 21-23 show the displayed information on both terminal and LCD display on several occasions such as alarm turning on (ThB > 37 °C) and back off as well as cover being opened and closed in our simulation.

A screen shot of a computer screen

Description automatically generated with low confidence

Figure 21: Alarm is on, Heater is off (PP = 1)

A screen shot of a phone

Description automatically generated with low confidence

Figure 22: Alarm is off, Heater is working according to PP

**A picture containing green, screenshot, font

Description automatically generated**

Figure 23: Changing cover status displayed on terminal

At this point, we saved our file as “Project\_Part7” to move onto the next part.

**Node-Red Implementation**

(i-Ege d-Ege t-Fatih e-Ege)

To communicate between Node-Red and Arduino, we used Virtual Serial Port Emulator (VSPE). First, we placed a COMPIM component on our schematic capture and connected IO0 and IO1 terminals to RXD and TXD ports respectively. We switched COMPIM’s physical port to COM10 and its baud rate to 9600. Then, in VSPE we created two connectors COM10 and COM11 as well as a serial redirector to connect these ports.

A screenshot of a computer

Description automatically generated

Figure 24: VSPE configuration

A picture containing text, font, number, line

Description automatically generated

Figure 25: COMPIM on schematic capture

After creating the ports, we launched Node-Red, which was previously installed on our computer by typing node-red command in command prompt and access to our canvas.

In Node-Red we placed a Serial-in node to receive inputs from Arduino. Each output node was connected to separate split nodes, followed by ThD, ThA and ThB connecting to charts and AA and Cover Status connecting to sliders to be displayed on Node-Red dashboard. For users to be able to change the settings of desired temperature ThD and proportional gain Kp on the Node-Red interface, we connected a serial out node to Kp and ThD sliders with respective functions given below (See Figure 26 for full design). Once the nodes were deployed and the simulation in Arduino was initialized, we saw that values were displayed on the dashboard as seen in Figures 27-30.

Function for Kp:

return { payload: "(".concat(msg.payload, ")p") };

Function for ThD:

return { payload: "(".concat(msg.payload, ")d") };

A picture containing text, diagram, line, screenshot

Description automatically generated

Figure 26: Design and outputs on Node-Red canvas

A screenshot of a computer

Description automatically generated with low confidence

Figure 27: Turned on alarm and temperatures on dashboard

A screenshot of a computer

Description automatically generated with medium confidence

Figure 28: Open cover and temperatures on dashboard

A screenshot of a computer

Description automatically generated with low confidence

Figure 29: Node-Red interface (Kp and ThD adjusted by user from sliders)

A screenshot of a computer

Description automatically generated with low confidence

Figure 30: Node-Red interface (Cover on)

**CONCLUSION**

**3.1. İbrahim Berk Alageyik**

As a team leader this project was certainly a vast experience for me. I had to put my skills as a leader to the test and organize and direct our group along the way to a successful end. I was responsible for the modelling analysis of our conceptual design part along with my friends Imad, Ahmed and **Isfendiyar. I learned how to create a conceptual design of a system, test the said system and conduct analysis.**

**3.2. Ege Barlas**

I was responsible from designing the hardware on Arduino and Node-Red. Even though we had our fair share of struggles we put great effort into it and hard work of the entire team led to completion of this project. It was one of the first times I’ve been in a group project that required this much effort and it was a great opportunity and experience for me, which I will benefit from in my remaining academic career as well as professional career ahead of me.

**3.3. Mohammad Imad Khan**

In conclusion, the aim of this project was to create a virtual infant incubator. Throughout the course of the project, I learnt about the workings of an embedded system and with the help of my teammates I believe we have made our project a success. My personal contribution to the project was the report and few of the SciLab simulation tests.

**3.4. Ahmet Fatih Saruhan**

I gained experience in this project, especially in the ARDUINO part. Together with my teammates, we tried to understand the logic of the components used, the layout of the circuit, the use of terminals and the management of the circuit with functions. I understood the importance of working as a team by acting together, correcting our mistakes and helping each other. We understood the importance of simulating a physical system on a computer. Thanks to my teammates and professor for their contribution.

**3.5. Emre Çakır**

In conclusion, this project has provided me with valuable experience, particularly in the ARDUINO part. Working collaboratively with my teammates, we focused on comprehending the component logic, circuit layout, terminal utilization, and circuit management through functions. Through our teamwork, we learned the significance of acting collectively, rectifying mistakes, and supporting one another. Additionally, we recognized the importance of simulating physical systems on a computer. I am grateful for the contributions of my teammates and professor, as their guidance was instrumental in our success.

**3.6. Ahmed Alghussein**

In conclusion, this was one of the best projects I've worked on. My team and I put a lot of effort into this project and used the tools that we acquired in this course in order to develop and construct this system. I was in charge of the overall view of the system and the simulation of the Embedded system, I also completed some documentation for the report. The project's overall task assignment idea is brilliant, and this was a truly wonderful experience because I got to work on something that would benefit me in the future. In closing, I want to express my gratitude to my professor, the lab assistants, and my friends for letting me participate in this fantastic experience.

**3.7. Ramazan Karataş**

During my participation in this project, we focused on developing a low-end control device utilizing Arduino technology. Our main objective was to ensure precise temperature control in the incubator environment for premature babies. Throughout the design phase, we paid meticulous attention to various aspects including hardware selection, sensor integration We implemented temperature measurements using thermistors, incorporated user input for customized settings, and controlled the heater to maintain the desired temperature. Furthermore, we tried display the real-time temperature readings on both an LCD display and the Node-Red platform for monitoring and adjustment. As a collaborative project team, we closely worked together on hardware design, software development, and seamless integration of all components. By emphasizing the critical importance of temperature control in the care of premature infants, this project has the potential to contribute significantly to advancements in the field.

**3.8. Isfendiyar Berkan Kasapoglu**

While working on this project was an interesting experience for a software engineer. Additionally, using tools like Scilab and Arduino was beneficial and positive for us. By adhering to the deadline and disciplinary guidelines, the entire team contributed to the project's completion. We completed our work in accordance with the directions we gave.  By working together as a team, we finished the job.  I am honoured to be involved in such a project. I want to express my gratitude to every one of my group members for their contributions.

**REFERENCES**

[1] M. Bodur, CMPE320 Embedded Systems Technical Requirements for IB04 Incubator Project, G-Magusa: M-Bodur, 2021

[2] M. Bodur, CMPE320 Embedded Systems Design Laboratory Sheet 5, G-Magusa: M-Bodur, 2021

[3] M. Bodur, CMPE320 Embedded Systems Design Laboratory Sheet 7, G-Magusa: M-Bodur, 2021

[4] M. Bodur, CMPE320 Embedded Systems Design Laboratory Sheet 6, G-Magusa: M-Bodur, 2021

[5] L. a. Seshia, Embedded Systems Design: Cyber Physical Approach, Boston: MIT Press, 2016.

[6] M. Bodur, CMPE320 Embedded Systems Design Technical Specifications for IncuTerm-1, G-Magusa: M-Bodur, 2021.

[7] *SciLab Documentation* (no date) *Documentation*. Available at: https://scilab.gitlab.io/legacy\_wiki/Documentation

[8] *Proteus tutorials* *PCB Tutorial Videos - Learn how to use Proteus EDA Tools*. Available at: https://www.labcenter.com/tutorials/