Temperature Controller Using Ardiuno with LabVIEW

Shivaji Lawate, Yash Bagewadikar and Rohit Misal

January 20, 2024

Temperature Controller using Arduino and LabVIEW

Shivaji C. Lawate, Yash R. Bagewadikar, Rohit P. Misal

Guide By:J.L.Musale

Abstract— The system uses Arduino and LABVIEW, as does the manual control system. We will be using an Arduino temperature controller that is LABVIEW based in this paper. When the temperature rises above the specified point, the 5-volt cooling fan in this paper is turned on. The LM35 temperature sensor will be used in this instance, and the temperature range will be indicated by Heater element. Three distinct categories will be used to classify the design and operation: first, when the temperature is above the mark (set) point, the heater element glow; second, when the temperature is between the higher mark point and lower mark (set) point, cooler if off; and third, when the temperature is below the set point, c00ler is off. Additionally, the temperature reading is kept in a Microsoft Excel document. Here, we designed the temperature controlling system using the LABVIEW 2017 software version. The other programs are also used to develop temperature controller systems, but LabView is the most straightforward of them all.

Key words: LABVIEW, Arduino, temperature, LM35, Heater element,peltier cooler module.

I. INTRODUCTION

The division of labor principle is seen in the way technology contributes to humankind by characterizing jobs that require only minimal effort from control systems. When an error occurs, the normal control system processes the controlled variable, correlates that dimension with the typical point or position, and then modifies its output signal to the manipulated variable. This nevertheless has enormous advantages because a relevant applied system is used to overcome a significant amount of strain and complexities, which can be tedious. Consequently, this concentrated on the domain of the temperature control system based on the control systems principle. The ideal plant temperature for the production and storage of goods is typically harmfully pretentious by too high or too low temperatures, which are difficult for conventional systems to adjust and control. As a result, an efficient and computerized system must grow in order to complete the task It is necessary to adhere to specific physical limits in a variety of sectors, including temperature, pressure, humidity, flow rate, etc. Different observation techniques are used to obtain detention measurements based on the quantities, distribution, and observed frequency of observed entities. The management choose whether to use an automated or manual monitoring system The terms "positive temperature coefficient" (PTC) and "negative temperature coefficient" (NTC) refer to the two types of temperature sensors. A thermistor sensor is significant because it is of the negative temperature coefficient of resistance (NTC) variety. Therefore, it continues to lose resistance as the temperature rises. As a passive transducer, this sensor requires an external power source to function.

Temperature control is important for separation and return procedures, and it needs to be maintained within certain bounds to ensure process equipment is safe and dependable. The process resources to check intervention with proper sensing and eliminate impairment to the sensor imperil the temperature sensor. Consequently, a barrier that is both chemically and physically resistant forms between the process and the sensor.

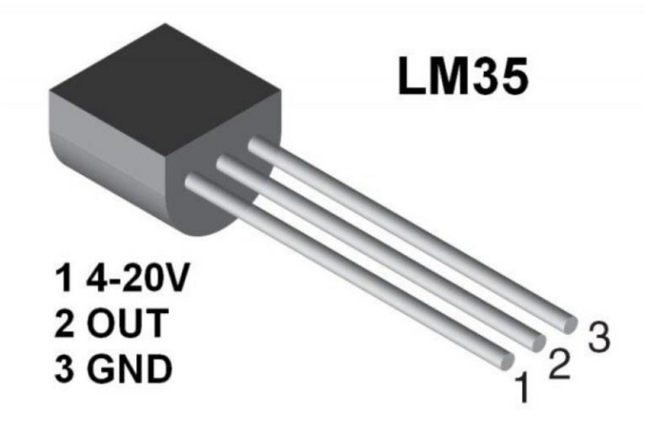


Figure 1: LM35 Temperature sensor.

The precision combined circuit temperature methods of the LM35 succession have an output voltage that is directly proportional to the temperature in centigrade. The LM35 temperature sensor is depicted in figure 1. It can provide traditional accuracy of ±1/4 °C at room temperature and ±3/4 °C over the entire temperature range of -55 °C to 150 °C without the requirement for any external adjustment or embellishment.

The left pin (pin1) of the LM35 temperature sensor is Vcc, the middle pin (pin2) connects to the analog pin A0, and the right pin (pin3) is ground.
Both software and hardware are involved in the planned effort.
Hardware: Arduino Uno Board, LM35 Temperature Sensor, peltier cooler module, Heater element.
Software: LabVIEW installed Laptop, LIFA, Arduino Software, MakerHub LabVIEW Packages.
The temperature sensed by the LM35 sensor, which is part of the executed system, is connected to LabView via MakerHub open LINX. LabView shows the variances in the temperature based on the measured value on the graph chart. When the temperature rises above the predetermined level, LabView instructs the Arduino Uno board to turn on the LED and activates the Boolean LED signal on the front panel of LabView.

II. HARDWARE DESIGN

Figure 2 displays the proposed system's block diagram. The laptop with LabVIEW installed, the Arduino Uno board, the LM35 sensor, Heater element, peltier cooler module, and connecting cables make up the suggested system implementation.

.

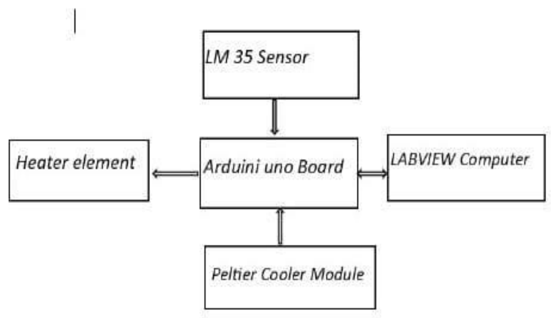


Figure 2: Proposed system block diagram

LM35 Sensor: In the suggested setup, the LM35 pin 1 is first linked to the power supply of the bread board and then to the 5 volt supply of the Arduino Uno. The Arduino Uno ground port is connected to LM35 pin3, while the analog A0 port is connected to LM35 pin

2.Arduino Uno Board:

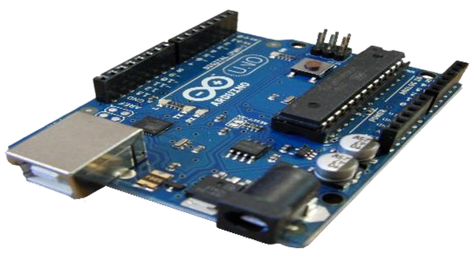
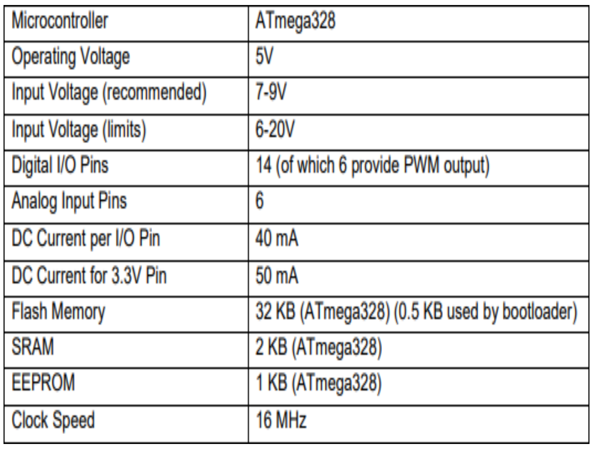


Figure 3: Arduino Uno Board

The Arduino Uno is a microcontroller board based on the ATmega328. The Arduino Uno consists of

- Input/output pins: 14 (of which PWM outpt pins: 6, Analog Input pins: 6).
- Ceramic resonator: 16 MHz
- USB connection
- Power jack
- ICSP header, and a reset button.



Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-9V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) (0.5 KB used by bootloader)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Table 1: Arduino Uno parameters.

Heater Element:

Description: The heater element serves as the heating component of the temperature control system. It is activated when the temperature falls below the predefined threshold, and it operates to raise the temperature within the controlled environment. The heater element provides a reliable means to maintain the desired temperature range.
Specifications:
Heating Power: Dependent on the specific heater element
Voltage Rating: Typically 12V
Current Rating: Dependent on specific heater element
Size and Dimensions: Varied based on the chosen heater element

Figure 4: Heater Element

Peltier Cooler Module:
A DC cooling fan with five volts is being used here. The fan is used to display the output of the system. Positive and negative terminals are present on this fan. The Arduino board's pin 3 is connected to the positive terminal, while the bread board's ground .s connected to the negative terminal. It outputs air flow as soon as it receives an input signal from the Arduino Uno board microcontroller.

When the temperature rises above a predetermined point, the fan terminal is connected to one of the LED logic pin designs in LabVIEW and is supported.

Figure 5: Peltier Cooler Module

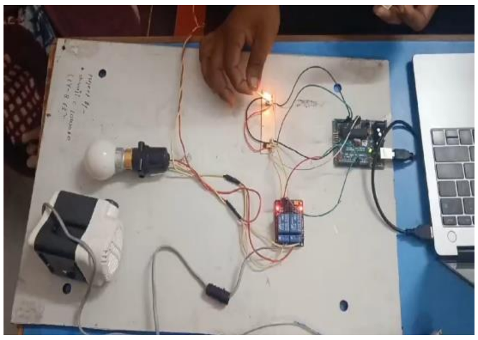


Figure 6: Hardware setup.

III. LABVIEW SOFTWARE DESIGN

To model the temperature from the building circuit divider and convert it into a temperature analysis in physical time, the first step in the software part is to program LabVIEW VI. The software that facilitates the interface between LabView and the actual Arduino hardware design is called LIFA, which stands for LabView interface For Arduino (or LINX).

We are able to both manage and see the evolution thanks to the front panel design. It has software inputs and outputs that mimic the actual controls of the hardware structure, including buttons, sliders, LEDs, speed controllers, and graphic charts showing temperature. The front panel design screenshot is displayed in Figure 7.

The Arduino Uno is connected to the LINX serial port; first, select the LINX MakerHub open. Next, choose the Whie loop. Within the loop, choose and drag the TMP35 sensor, which is attached to the input of the LINX resource as the output.

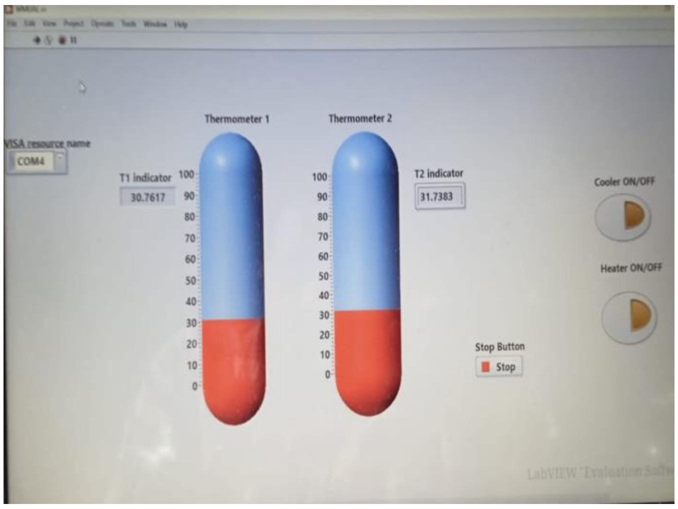


Figure 7: Temperature controller Front panel

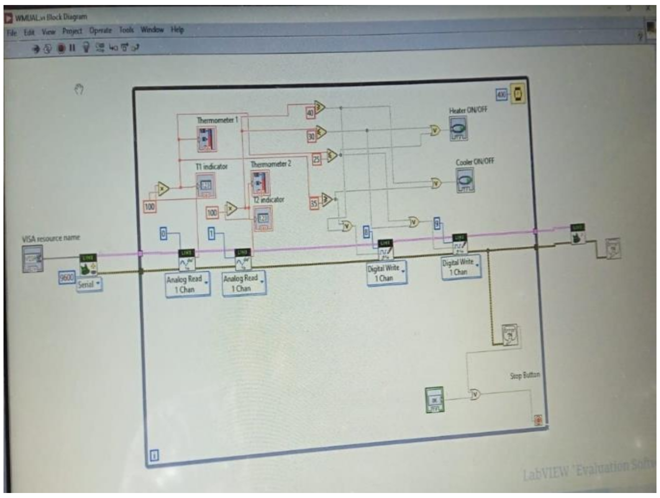
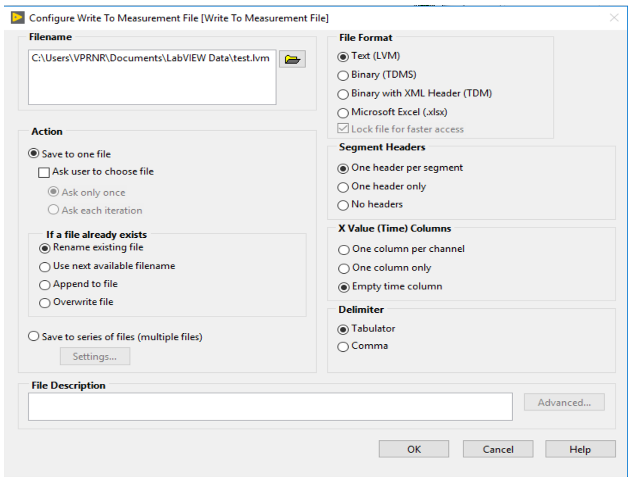


Figure 8: Temperature controller Block diagram panel



Configure Write To Measurement File [Write To Measurement File]

Filename

C:\Users\VPRNR\Documents\LabVIEW Data\test.lvm

Action

● Save to one file
☐ Ask user to choose file
● Ask only once
○ Ask each iteration

If a file already exists
● Rename existing file
○ Use next available filename
○ Append to file
○ Overwrite file

○ Save to series of files (multiple files)
Settings...

File Format
● Text (LVM)
○ Binary (TDMS)
○ Binary with XML Header (TDM)
○ Microsoft Excel (.xlsx)
✓ Lock file for faster access

Segment Headers
● One header per segment
○ One header only
○ No headers

X Value (Time) Columns
○ One column per channel
○ One column only
● Empty time column

Delimiter
● Tabulator
○ Comma

File Description

Advanced...

OK Cancel Help

Figure 9: Write to measurement file functional palette.

The labview block "Write To Measurement File," which stores the various temperature reading results that are exported into Microsoft Excel, is connected to the sensor's output. The suggested system's connecting procedure is depicted in figure 8.

IV. ARDUINO INTERFACE WITH LABVIEW (LIFA AND LINX)

How to Interface Arduino with Labview:

following block diagram panel design and front panel design. We need to install the LINX device driver before we can interface the Arduino Uno microcontroller port. LabView uses either LINX or LIFA to communicate with the Arduino Uno. The LabView section displays the temperature sensor value. By using the install drive driver to retrieve the embedded program's temperature sensor, the LabView portion alters the connection. The driver installation procedure is followed by the steps listed below.

Step 1: first, use the VI package manager to obtain LIFA and the MakerHub interface.

Step 2:An error such as "VIPM could not continue" will appear once it is finished..

Step 3:Launch LabVIEW. Choose Tools, then click the choice, and finally click VI server.

Step 4:Move down to verify the machine access address after making progress with the VI Server, and then manually add the machine access list. Verify that you have "127.0.01," "localhost," and "*" added to the Machine access list before clicking "OK."

Step 5: Now open downloaded VI Package Manager.

Step 6: Search for LabVIEW Interface for Arduino and double click on it to install it.

Step 7: Click on continue and after installation is, completed click on finish.

Following LIFA completion, a 2 Hz loop rate selection is made, with the maximum value of 35 and the lowest value of 25 displayed in Figure 10. Navigate to Tools → MakerHub → LINX → LINX Firmware wizard before launching the LabView software. Click the Next button after selecting the Arduino Device Family, Arduino Uno Device Type, and Serial/USB Firmware Upload Method in the LINX Firmware Wizard. Next, choose the Arduino Uno port (COM13), and then click the Next button once more. Lastly, press the "Finish" button. In figure 11, this procedure is displayed. If not, it will show the error below.

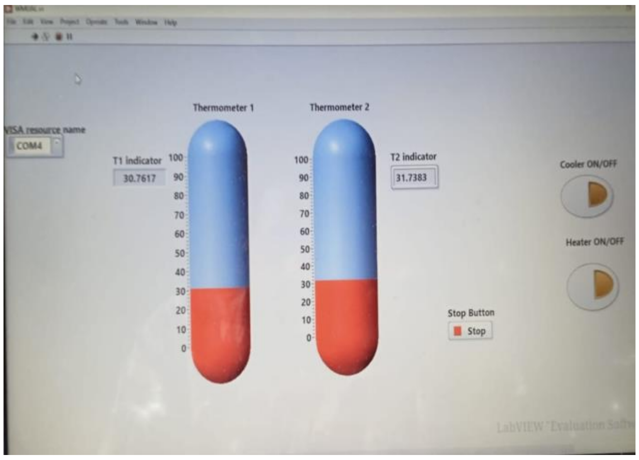
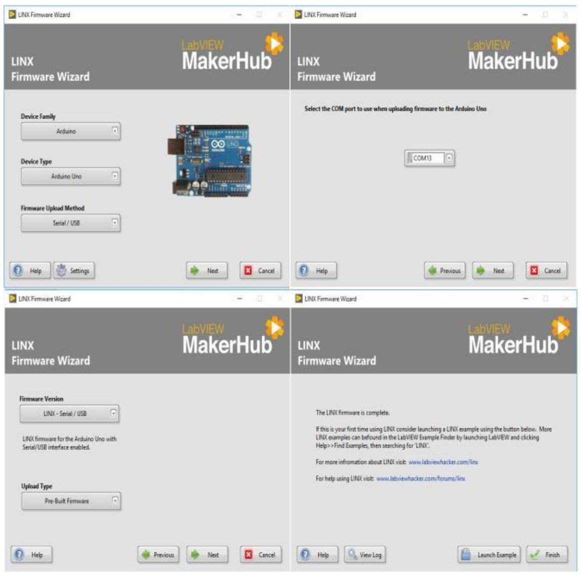


Figure 10: Temperature Controller setup values.



Device Family: Arduino
Device Type: Arduino Uno
Firmware Upload Method: Serial / USB

Select the COM port to use when uploading firmware to the Arduino Uno
COM13

Firmware Version: LINX - Serial / USB
LINX firmware for the Arduino Uno with Serial/USB interface enabled.
Upload Type: Pre-Built Firmware

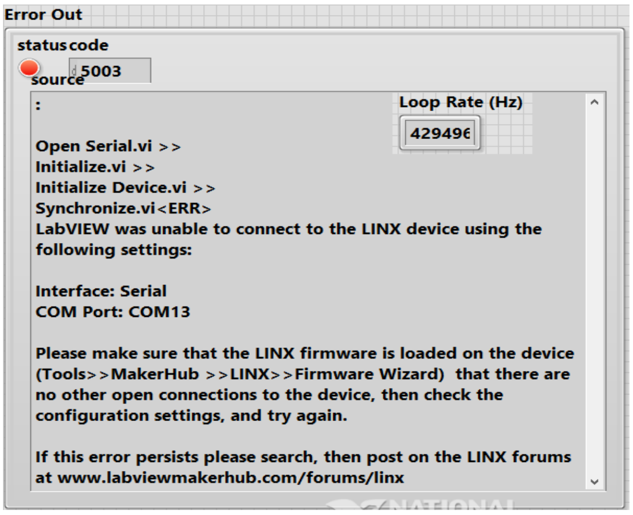
The LINX firmware is complete.

If this is your first time using LINX consider launching a LINX example using the button below. More LINX examples can befound in the LabVIEW Example Finder by launching LabVIEW and clicking Help>>Find Examples, then searching for 'LINX'.

For more infromation about LINX visit: www.labviewhacker.com/linx

For help using LINX visit: www.labviewhacker.com/forums/linx

Figure 11: LINX MakerHub Process.



```
Error Out
status code
source 5003
:                                   Loop Rate (Hz)
                                    429496

Open Serial.vi >>
Initialize.vi >>
Initialize Device.vi >>
Synchronize.vi<ERR>
LabVIEW was unable to connect to the LINX device using the
following settings:

Interface: Serial
COM Port: COM13

Please make sure that the LINX firmware is loaded on the device
(Tools>>MakerHub >>LINX>>Firmware Wizard) that there are
no other open connections to the device, then check the
configuration settings, and try again.

If this error persists please search, then post on the LINX forums
at www.labviewmakerhub.com/forums/linx
```

V. RESULTS AND DISCUSSION

After effective design and implementation of proposed system and examination run, the output product displays when the factories situation temperature is changing at 34^0C, although the temperature range condition is between 25^0C to 35^0C, so thecooling fan system unit does not run.

Here, we set the maximum temperature 35^0C, if it exceeds Heater element will glows on LabView front panel and hardware portion. Now, we are going to close my temperature sensor, the Thermometer reading exceeds the maximum temperature,we can also see that the fan is started, which is shown in figure 12. Figure 12 shows the output result when the factories setting temperature is fluctuating at 38^0C, while the requirement of temperature range is 25^0C to 35^0C.

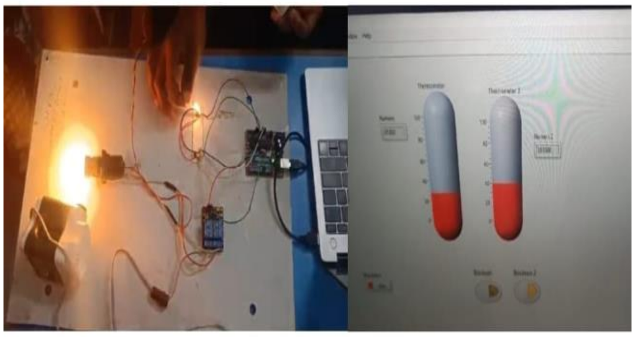


Figure 12: hot temperature

Now, we are removing fingers from my temperature sensor, the temperature is gradually reducing and cooling fan also turned OFF.

Figure 13 shows the output result, when the factories setting temperature is changing at 35^0C, while the requirement range of temperature is between 35^0C to 40^0C, so the cooling fan system unit does not run in this case.

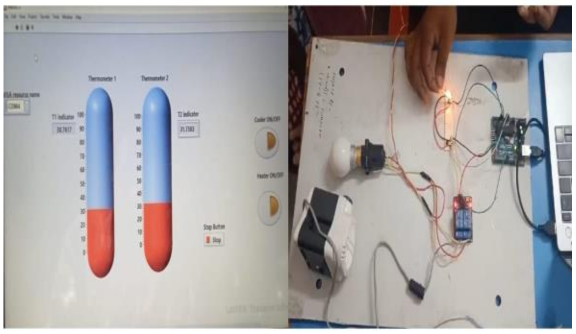


Figure 13: moderate temperature

Table 2: Data obtain during test

S.NO	DATE	TIME	TEMP ^{0}C
1	21-11-2023	19:37:06.543	33.691406
2	21-11-2023	19:37:14.120	34.179687
3	21-11-2023	19:37:31.509	33.691406
4	21-11-2023	19:37:43.501	34.179687
5	21-11-2023	19:39:43.582	37.109375
6	21-11-2023	19:39:54.588	38.574219
7	21-11-2023	19:40:33.585	36.621094
8	21-11-2023	19:42:34.280	35.15625
9	21-11-2023	19:46:00.989	35.644531

CONCLUSION

This paper's primary goal is to store and present data on LabView while controlling the industrial environment's temperature both manually and automatically.

1. The cooling fan system unit does not operate when the factory temperature changes to 340C, even though the temperature range is between 250 and 350C. This is indicated by the output product. On the LabView front panel and hardware setup,

2.The end product when the factory temperature is constantly shifting between 380C and the required temperature range of250C to 350C. Consequently, the cooling fan system unit turned on by itself. The Heater element is glowing on LabView front panel and Hardware setup.

3. The cooling fan system unit does not operate in this situation because the factory temperature setting is changing at 350C, even if the required temperature rang e is between 350C and 400C. On the LabView front panel and hardware Setup.

We also recommend upcoming design to include other structures such as air condition and weather station controllerand advance sensing unit.

REFERENCES

1.Jim Baker.How to call win32 Dynamic link Libaries(DLLs) fromLabVIEW. National Instruments Notes 088. www.ni.com

2.Based on NImyDAQ Sound Processing System, College of Electronics and Information Engineering, Changchun University,Changchun,china.

3.Using LabVIEW to Measure Temperature with a Thermistor,C. Briscoe and W. Dufee3., University of Minnesota.

4.P.A.Harsha Vardhini, Y.Murali Mohan Babu, A.Krishna Veni, “Industry Parameters Monitoring and Controlling System based onEmbedded Web server”, International Journal of Emerging Technologies and Innovative Research, Vol.6, Issue.2,page no. pp.80-84, February 2019.

5.Real Time Temperature Monitoring Using LABVIEW and Arduino, Vaibhav M. Davande1, Pradeep C. Dhanawade2, Vinayak B. Sutar3, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 4, Issue 3, March 2016.

6.Rajesh, S. K. (2013), ‘‘Industrial Temperature Monitoring And Control System through Ethernet LAN ’’International Journal of Engineering and Computer Science ISSN: 2319-7242 Volume 2 Issue 6 Page No. 1988-1991.

7.K. Chenna Kesava Reddy, P.Venkatrao, “Real Time Field Monitoring and Controlling System”, International Journal of Recent Technology and Engineering (IJRTE), Volume-2, Issue-4, pp.-98-100, September 2013.

8.N.Koteswaramma, P.A.Harsha Vardhini, “Implementation of Arduinobased Object Detection System”, International Journal Of Modern Electronics and Communication Engineering (IJMECE),pp.2018-211,Vol. 7, Issue.3, May 2019.

9.K. Murali Chandra Babu, P. A. Harsha Vardhini, N.Koteswaramma, “Design and Implementation of Arduino Based Riders Safe Guard 2.0” ,International Journal of Innovative Technology and ExploringEngineering (IJITEE), pp.3078-3083, Vol.9 , Issue.1, Nov 2019.