Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

Control Board of Commercial Stove

Milav Soni
M. Tech (Embedded System and VLSI Design)
C.S.P.I.T, CHARUSAT University
Changa, Gujarat, India

Hitesh Patel
Faculty of Electronics & Communication
C.S.P.I.T, CHARUSAT University
Changa, Gujarat, India

Mr. Siddharth Mistry
Product Engineer,
Teq Diligent Product Solution Pvt. Ltd,
Ahmedabad, Gujarat, India

Jignesh Patoliya
Faculty of Electronics & Communication
C.S.P.I.T, CHARUSAT University
Changa, Gujarat, India

***Abstract*: The Control board provides control for fuel feeding and burning the pallets used in company. The single control board can control clean energy stoves of Model-1. It has interactive user interface to control and monitor various fuel feeding and burning requirements. The Model-1 product has automatic control mechanism based on temperature input from the cooking vessel. Thus a model-1 is proposed which utilized multi layer approach along with principle components analysis. The qualitative and quantitative results show that the proposed model gain advantages of lower complexity along with approximately 90% accuracy.**

Keywords—Controller(8-bit), Thermocouple, GSM Module, Serial Protocol (UART and SPI)

I. INTRODUCTION

The main aim of this Commercial Stove to provide a biomass pellet based large scale cooking solution. In user point of view, this Commercial Stove provide saving against Gas and Diesel up to around 30%. This Commercial Stove has two models (1) Model-0.

This Model is uniquely designed for large scale continuous cooking application, using a clean, cost effective and eco-friendly fuel. It is an energy efficient appliance that is an ideal alternative to fossil fuels such as a LPG, PNG, and Diesel.

As shown in figure1 the Control board is an electrical hardware device containing equipment for regulating electrical devices that makes decisions regarding whether or not proposed changes to a software project should be implemented.

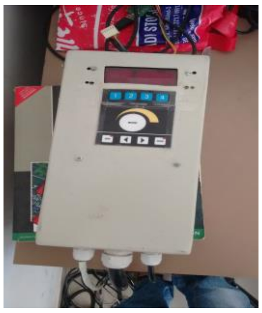


Fig. 1 Control Board

II. WORKING OF COMMERCIAL STOVE

As Shown in figure2, the pellets are put inside the chamber1. And heater is put inside the chamber2.

When power is turn on and user press the Ignition key than motor and heater are turn on and pellets are move to chamber2 via belt and due to heater on the pellets are burn. After completion of the ignition process the user press normal button. In that motor will be on and off and two fans are on continuously as per defined speed. Due to this flame will be created and cooking will be start at this mode. Thermocouple sense the temperature of cooking plats and using that value the stove will change mode automatically.
This stove has two modes.

1) High Mode
2) Low Mode (By Default run this mode)

The value of motor on/off time, heater on time, fans speed is different in these two modes. The value of motor on/off time and some other feature will decided by user and it is re-configurable by user.

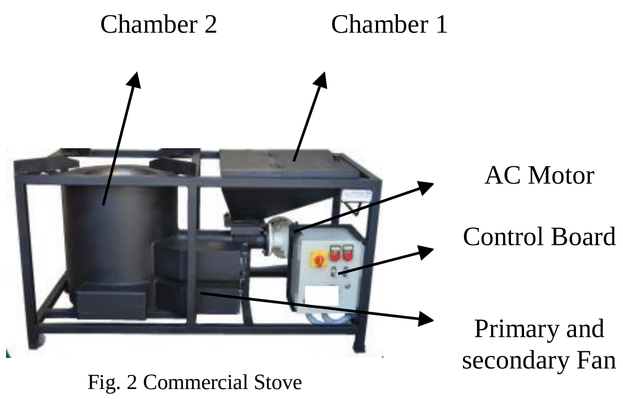


Fig. 2 Commercial Stove

Figure3 shown block diagram of control board. This control board consists two main Microcontrollers with programmable input/output peripherals.

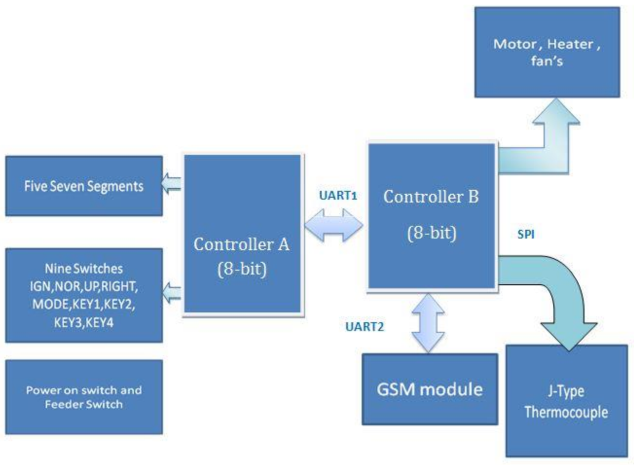


Fig. 3 Control Board Block Diagram

III. WORKING OF CONTROL BOARD

The Control Board is run on the following four modes.
<u>Ignition Mode</u>:-In this mode the control board turns on the heater GPIO and Motor GPIO. This on time is pre-stored in the microcontroller. This time is reconfigurable by user.
<u>Normal Mode</u>:-In this mode the control board turns on the Fans GPIO and Motor GPIO. Motor will be turn on/off at pre-defined time and fan will on at pre-defined duty cycle (speed).This time is pre-stored in the microcontroller. This time is reconfigurable by user.
<u>Configuration Mode</u>: - This mode is new feature in the Commercial stove. Using this mode we can change the pre-stored value of some new function as per requirement of user by press some couple of switches. This mode is run parallel to above two modes.
<u>Debug Mode</u>: - This mode is also run parallel to the first two modes. This mode displays the motor on and off time value on seven segments. Using this mode user can debug the motor time data.

First two modes are run sequential and last two modes are run parallel to above two modes.

A. *Working of Switches and Seven Segments:-*

We have total eleven switches are use in the controller board. The Sample of Membrane Switch is shown in figure 4.
<u>Mains</u>: - it is used to start the Control board.
<u>IGN</u>: - By press this switch board can enter in **Ignition Mode**. And Ignition switch led will be on and "Ign" will display on seven segments.
<u>NOR</u>: - By press this switch board can enter in **Normal Mode**. And switch led will be on And "nor" will display on seven segments. Press second time the normal mode will be off and led will be off.
<u>UP (<)</u>: - for changing any segment value user press this switch.
<u>Right (>)</u>: - for changing any segment position (cursor) user press this switch.
<u>Up+Right (< + >)</u>: - when user presses both this switch at a time than Board will go into **Configuration Mode**. And "Fn---" will display on seven segments.
<u>IGN+NOR</u>: - when user presses both this switch at a time than Board will exit **Configuration Mode**.
<u>KEY1</u>: - By press this switch board can enter in **Debug Mode**. And Display on/off time on seven segments.
<u>KEY3</u>: - By presses this switch board will send message to user through GSM module.

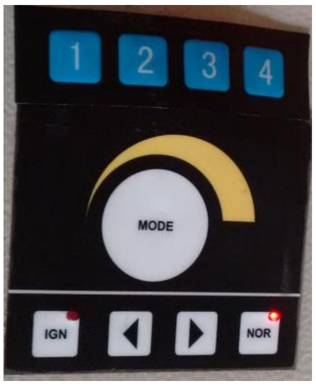


Fig. 4 Membrane Switch

B. *Working Of UART 1 Protocol:-*

This Control board consist two microcontrollers interfacing Using UART1 protocol. We first see this communication to understand other concepts.

For synchronization between these two microcontrollers we use some pre-defined command based UART communication.

We use some formats for better and reliable communication between these controllers.
Both controllers have transmitted some value and received some value. To avoid overwrite these two value we use one special end character **dollar** (**$**). This character indicates the ending of one value and starting of another value. One of that controllers send some value with last one "**$**" characters.
I.e. A Controller send "**24$**" value to B controller. At the other hand B controller received the value from A Controller until "**$**" received.

Using this format we cannot lose any value and implemented proper sync communication. We see this command one by one.

- **34$**:- A controller sends this value to B controller for initialization purpose. The B controller waits until received this value. After getting this value B controller execute next step. The second purpose of this command is to recover the last step of A controller. i.e. if A controller is go into reset state than it send again this command to B controller and after getting this command the A controller send running process using next command.

- **35< latest mode>$**:- When A controller go into reset state than B controller send "35 with running mode (IGN/NOR) and last $". So that A controller recover the state.

- **35$**:- When B controller go into reset state than it's send this command and replay of this command A controller send appropriate command. I.e. if ignition will be start than 00$ etc.

- **00$**:- When user press the IGN switch than A controller send this command to B controller. After receiving this command B controller turn on Heater and motor GPIO. And "**IGN**" will be displayed.

- **57$:-** This on time is stored in B controller. After complete this time the GPIO will be off and B controller send this command to A controller to display the "**rdy**" on seven segment.

- **0100$:-** When NOR key is press than A send to B controller for starting the normal process. After getting to this value B controller start the fan and motor GPIO. And "**nor**" display.

- **0103$:-** When second time NOR key is press than A send to B controller for terminate the normal process. After getting to this value B controller turn off the motor GPIO and remain on fan GPIO for pre-defined time. And "**OFF**" will be display on seven segments.

- **02$:-** When user presses two key (< + >) at a time than board go into Configuration mode and A controller send this command to B controller and display "**Fn---**".

- **03<Disp5><Disp4>$:-** In configuration mode when user set the function through UP and Right key than A controller send this command to B controller. This command is send after pressing MODE key.

- **04<Disp3><Disp2><Disp1>$:-** In configuration mode when user set/change function value or not when second time MODE key is press than A controller send this command with segment value.

- **52<Disp5><Disp4><Disp3><Disp2><Disp1>$:-** After receiving upper both command the B controller send this command to the A controller for display purpose.

- **05$:-** When user presses two key (IGN+NOR) at a time than board will go out from configuration mode and A controller send to B controller this command.

- **06$:-** When user press KEY1 first time than A controller send this command to B controller to start debug mode.

- **60<Disp-5><Disp-4>$:-** After receiving 06$ the B controller send motor on/off count value as two digit format using this command. This command received by A controller display that value on seven segment.

- **07$:-** When users press KEY1 second time than A controller send this command to B controller to stop debug mode.

- **33<Disp3><Disp2><Disp1>$:-** When board is in normal mode its display temperature every 5 second. We have added some new feature in this normal mode. User set the temperature value direct from screen without going in to configuration mode. This can be achieved by send this command to B controller from A controller by pressing UP and Right Key format. In response of this command B controller set this value in that memory. And changes are made according to temperature value.

- **54<Disp5><Disp4><Disp3><Disp2><Disp1>$:-** In configuration mode when user set the function 15 than A controller send "0315$" to B controller in response of that command B controller send 5 Digits total **Machine Hour value** to the A controller for display purpose.

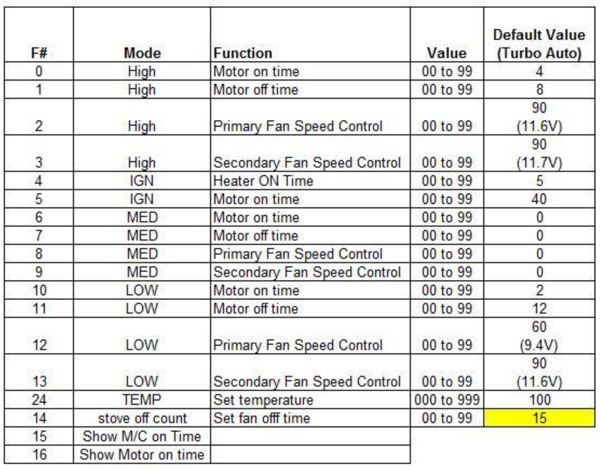
- **55<Disp5><Disp4><Disp3><Disp4><Disp3>$:-** In configuration mode when user set the function 16 than A controller send "0316$" to B controller in response of that command B controller send 5 Digits total **Motor Hour value** to the A controller for display purpose.

C. Working Of EEPROM:-

We have total 512 Bytes EEPROM Memory available in controller-A [2]. The basic use of this EEPROM is to stores factory default value of the Stove. And user can change that factory value in configuration mode. The default factory default values are stored in main memory and EEPROM memory.

As shown in figure5 we have total 15 functions which have default value are stored in EEPROM.

D. Working Of Thermocouple:-



F#	Mode	Function	Value	Default Value (Turbo Auto)
0	High	Motor on time	00 to 99	4
1	High	Motor off time	00 to 99	8
2	High	Primary Fan Speed Control	00 to 99	90 (11.6V)
3	High	Secondary Fan Speed Control	00 to 99	90 (11.7V)
4	IGN	Heater ON Time	00 to 99	5
5	IGN	Motor on time	00 to 99	40
6	MED	Motor on time	00 to 99	0
7	MED	Motor off time	00 to 99	0
8	MED	Primary Fan Speed Control	00 to 99	0
9	MED	Secondary Fan Speed Control	00 to 99	0
10	LOW	Motor on time	00 to 99	2
11	LOW	Motor off time	00 to 99	12
12	LOW	Primary Fan Speed Control	00 to 99	60 (9.4V)
13	LOW	Secondary Fan Speed Control	00 to 99	90 (11.6V)
24	TEMP	Set temperature	000 to 999	100
14	stove off count	Set fan offf time	00 to 99	15
15	Show M/C on Time			
16	Show Motor on time			

Fig. 5 Function Table

Thermocouple is used for read the temperature of stove and according to this stove will be on and changes their modes. Here we design only model-1 of stove so we have only two modes are applicable. One is HIGH Mode and second is LOW Mode. Both modes have different value as shown in figure20.

Here thermocouple has two wires T+ and T-. These two wires cannot connect directly to the GPIO. For that we used some thermocouple to digital converter device. For that we use **MAX31855 IC** for sense the digital value of thermocouple temperature value.

The MAX31855 performs cold-junction compensation and digitizes the signal from J-Type Thermocouple [3]. The data is output in a signed 14-bit, SPI-Compatible, read only format [3]. This IC detects Thermocouple error like, Open thermocouple, thermocouple short to Ground and power supply [3].

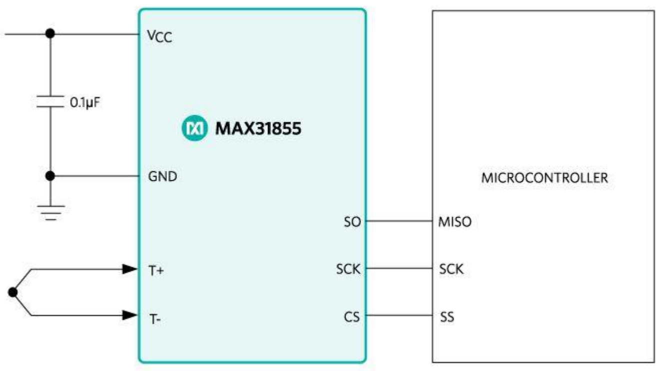
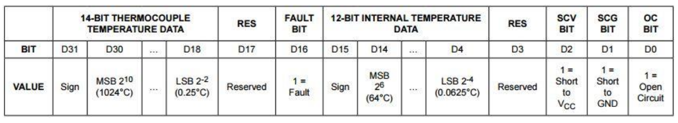


Fig. 6 Thermocouple connection

Figure 6 [3] show the SPI connection between Max31855 and Controller A controller. From this IC we can also get internal Board temperature.

Controller A use SPI-Read format to read the digital temperature. This IC supply 32-bit digital value including External temperature + internal temperature and error bit as shown in figure 7 [3].



	14-BIT THERMOCOUPLE TEMPERATURE DATA				RES	FAULT BIT	12-BIT INTERNAL TEMPERATURE DATA				RES	SCV BIT	SCG BIT	OC BIT
BIT	D31	D30	...	D18	D17	D16	D15	D14	...	D4	D3	D2	D1	D0
VALUE	Sign	MSB 2^{10} (1024°C)	...	LSB 2^{-2} (0.25°C)	Reserved	1 = Fault	Sign	MSB 2^{6} (64°C)	...	LSB 2^{-4} (0.0625°C)	Reserved	1 = Short to V_{CC}	1 = Short to GND	1 = Open Circuit

Fig. 7 Temperature Format

For getting the 32-bit value Controller A send 32 clocks continuous to max31855 IC. After getting the clocks max31855 IC send 32 bit value to the Controller A controller as shown in figure 8.

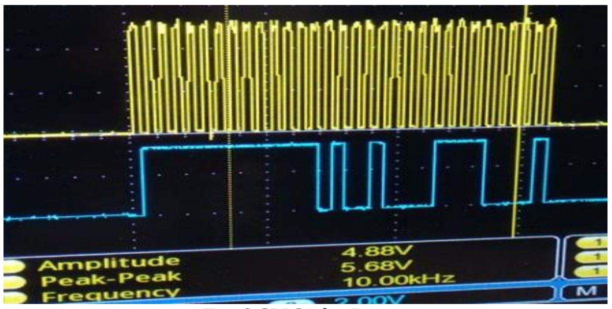


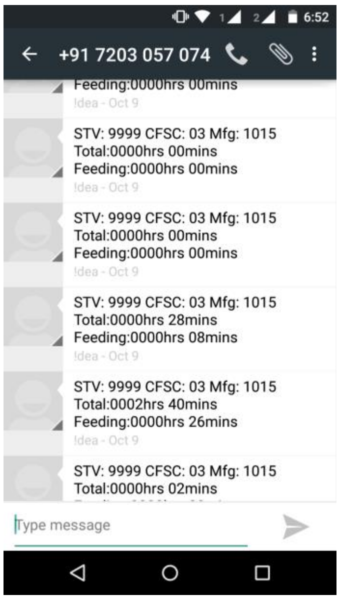
Fig. 8 SPI 32-bit Data

E. *Working Of UART 2 Protocol:-*

This UART protocol is used for GSM communication with control board. GSM module is used for send the motor on hour and machine on hour's value to the user mobile.

For that we use one switch KEY3 to send the data via GSM communication. This data have some special format as shown in figure 9.

Total indicates total machine hours and minute value. **Feeding** indicates total Motor on hour and minute value. For AT command we refer simply to the reference number, as in [4]



6:52
+91 7203 057 074
Feeding:0000hrs 00mins
Idea - Oct 9
STV: 9999 CFSC: 03 Mfg: 1015
Total:0000hrs 00mins
Feeding:0000hrs 00mins
Idea - Oct 9
STV: 9999 CFSC: 03 Mfg: 1015
Total:0000hrs 00mins
Feeding:0000hrs 00mins
Idea - Oct 9
STV: 9999 CFSC: 03 Mfg: 1015
Total:0000hrs 28mins
Feeding:0000hrs 08mins
Idea - Oct 9
STV: 9999 CFSC: 03 Mfg: 1015
Total:0002hrs 40mins
Feeding:0000hrs 26mins
Idea - Oct 9
STV: 9999 CFSC: 03 Mfg: 1015
Total:0000hrs 02mins
Type message

Fig. 9 Message Format

IV. POJECT ISSUES

A. *Hardware Issues:-*

- In board I solder wrong position of Transistor TIP122 (PWM generate Transistor) i.e. I connect emitter in base pad. So due this connection transistor cannot generate proper output i.e. PWM frequency.
- Motor and Heater relay transistor also connect in wrong position. So due to this relay cannot operate properly.
- Due to damage of heater control fan transistor the second fan will always on condition. So I can change that transistor.
- In ac supplies I connect only AC relay connection. But using that there are many spike are come in ac

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

supplies so that motor will be on/off as per spike. The solution of this problem use sunbber circuit replace with mechanical relay.

- Due to above problem there are also spike in 5v and 12v supply so that 12v IC and 5v IC cannot also work properly. This problem cause by motor on and off process continuously. And these recover by using capacitor across supply voltage.

B. *Software Issues:-*

- In program I use PWM frequency 31khz which is more for our application so that I use prescaler 8 for generate around 3khz frequency. Due to this problem I cannot generate proper speed of particular voltage Fan.
- In SPI programming I send continues 8 clocks in four time with some delay between them. Due to this problem I cannot read MOSI data from slave. The solution of this problem is send 32 clock continuously.
- At prior I use internal 8 MHz frequency for timer 2 ISR. Using that for generate 1 second loop-ISR, I use one count variable which have a value 31 which is approximant. So I cannot generate exactly 1 second delay. So instead of that I use external 32.767 kHz crystal frequency which work like a RTC and generate exactly 1second delay which I need.
- In temperature reading logic I cannot implement hysteresis logic. Due to this problem the transition between high mode to low mode and vice versa is not work properly. The solution of this problem is implemented PID logic in Temperature reading.

V. CONCLUSION

This paper provides control for fuel feeding and burning the pallets used in Commercial stove products. The single control board can control clean energy stoves. In this paper also discussed about PID logic implement in sensing the temperature value to prevent from Hysteresis loop. In this paper also discussed the commercial stove run in different mode for improving the performance.

REFERENCES

[1] http://www.atmel.com/Images/doc1919.pdf
[2] http://www.atmel.com/Images/Atmel-8011-8-bit-AVR-Microcontroller-Controller A-324P-644P_datasheet.pdf
[3] https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/MAX31855.html
[4] http://www.amuroboclub.in/downloads/ebooks/GSM%20MANUAL_AT%20COMMANDS_SIM900_ATC_V1_00.pdf
[5] Shi Trianyun, "Research and development of integrated intelligent control system of cooking stove combustion". In IEEE, 2002, 2347-2352,vol3.
[6] Riely p. h,"Designing a Low-Cost, Electricity-Generating Cooking Stove". In IEEE, Summer 2010,47-53.
[7] Sun jinsheng "Case-Based Real-Time Controller and its Application in Combustion Control of Hot Blast Stoves". In IEEE,7792-7796.
[8] Smith, "Application and limitation of Thermocouples for measuring temperatures". In IEEE, Aug. 1923, 851-853.
[9] Anan Fang, "Using Operational Amplifiers to Realize the Compensation of thermocouples". In IEEE, 3-4 Aug.2008, 774-777.
[10] Bajzek, T.J.,"Thermocouples: a sensor for measuring tempratur". In IEEE, March 2005, 35-40.
[11] Correa, C.R. "Embedded controller software and algorithm development tool". In IEEE, June 2003, 885-890.