Department of Electronic & Telecommunication Engineering

University of Moratuwa

Hearth

Group EN09

200222K	HIMEKA S.H.D.
200232P	INDRAPALA S.A.
200239T	JATHURSHAN S.
200242V	JAYAKODY J.A.A.U.

This report is submitted for module EN1190

07th October 2022

Abstract

The hearth regulates the temperature of the fire as per the user requirement by using the reading taken from the temperature sensor and regulating the DC fan speed accordingly. The hearth has a user interface which allows the user to turn a knob to their required temperature range and the selected range will be displayed on the OLED. Additionally, the hearth also has a buzzer going off if there are any abnormalities when functioning. The enclosure for the hearth was designed using SolidWorks and attention was given to the materials employed. The PCB was designed using Altium.

Contents

Introduction

The hearth takes into consideration the input temperature range by the user as well as the current temperature reading from the temperature sensor and will change the fan speed in order to bring the temperature into the users' temperature range. The main components of the device are,

- ATmega328P-PU
- MAX6675 Temperature Sensor
- OLED
- 100K Potentiometer
- 12V DC fan
- L9110 Motor Driver Module
- 12V Power Pack

The device gives the user an option of 6 temperature ranges and has a buzzer and a display to notify the user of the selected temperature range and if there is an issue in the functionality.

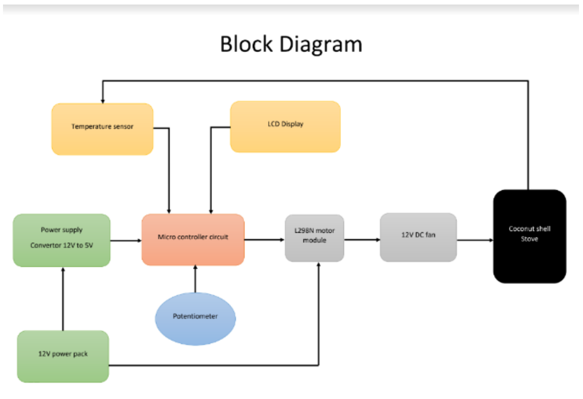


Figure 1.1: Block Diagram of the Device

Product Description

Design Idea

Given the shortage of gas cylinders and the hike of gas cylinder prices in the recent past in Sri Lanka, most people have been turning to using electric induction cookers or hearths. Since electricity prices as well as induction cooker prices have also increased, most households have been using traditional hearths.

Motivation

However, these hearths have potential to be improved to make the cooking experience more user friendly. The existing market products can be seen to lack the following features,

- User must keep blowing into the flame to control the flame. Even the hearths with DC fans have an unreliable increase and reduction in temperatures such that the user must consistently be present to adjust fan speeds.
- Soot depositions on cooking utensils and the stove.
- Cleaning the ash leftovers after usage.
- Smokes released are disturbing.

Who Benefits?

Our device would cater to most lower to middle class households as it could reduce the costs associated with gas cylinders as well as electricity.

Product Idea Validation

Hearths have a lot of room for improvement. However, the main problem we chose to focus on was the fact that the user must be in constant vicinity of the hearth to ensure that the heat from the fire is regulated at the needed level. This would be a waste of time for the consumer.

Therefore, our solution was to build a hearth with a DC fan which would regulate the heat in the needed level once the user has input their requirement.

To ensure that this was a requirement of our target market we conducted a survey through which we collected the following data.

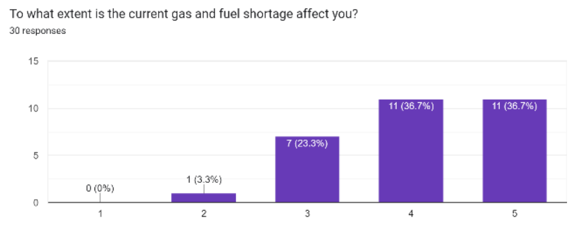


Figure 3.1: The extent to which users were affected by the gas shortage.

As seen by Figure 3.1, more than 50% of users were highly affected by the gas shortage. Therefore, an alternative for gas cookers is necessary.

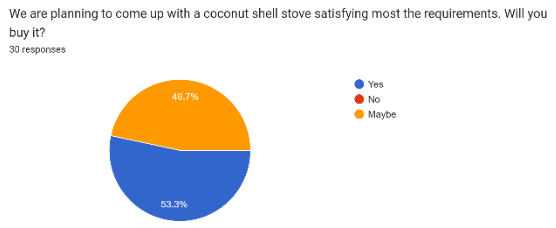


Figure 3.2: Will the user buy our product?

On explaining the product, we had in mind, most users showed positive responses whereas a few were unsure but none of the users believed that the product was not needed.

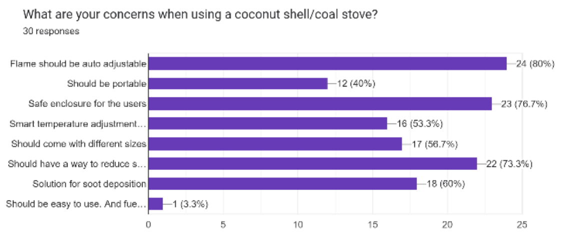


Figure 3.3: User needs.

We then took this opportunity to check what the user expected of this product and on designing the product we took into consideration most of these factors.

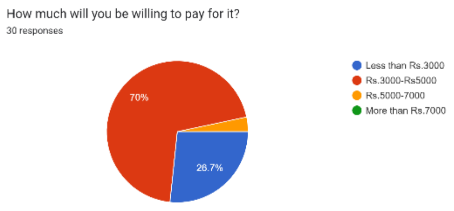


Figure 3.4: Product cost

Finally, we asked the users what cost they would be willing to incur for this product. As per Figure 3.4, the price range was determined to be between Rs 3,000 and Rs 5,000. However, this survey was conducted approximately 2 months prior and did not take into consideration the import bans as well as the inflation. Currently, the market price for similar products range between Rs 7,000 and Rs 11,000. Therefore, the price for the product may have to be reconsidered to around Rs 8,000 or more.

Development

The development of this project was done in four stages. They are,

1. Circuit Design
2. Arduino Implementation
3. Breadboard Implementation with the Microcontroller

Circuit Design

Initially a schematic with all the circuit components was built using the EasyEDA software. Afterwards, the connections between each component were made.

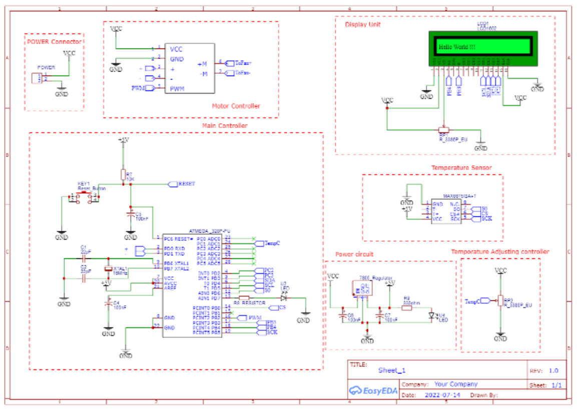


Figure 4.1: Circuit components

Arduino Implementation

Once all the circuit connections were made, the circuit was implemented on an Arduino step by step to check its functionality.

Initially, instead of using a DC fan, we placed LED bulbs in order to check whether the brightness can be changed for different temperature levels.

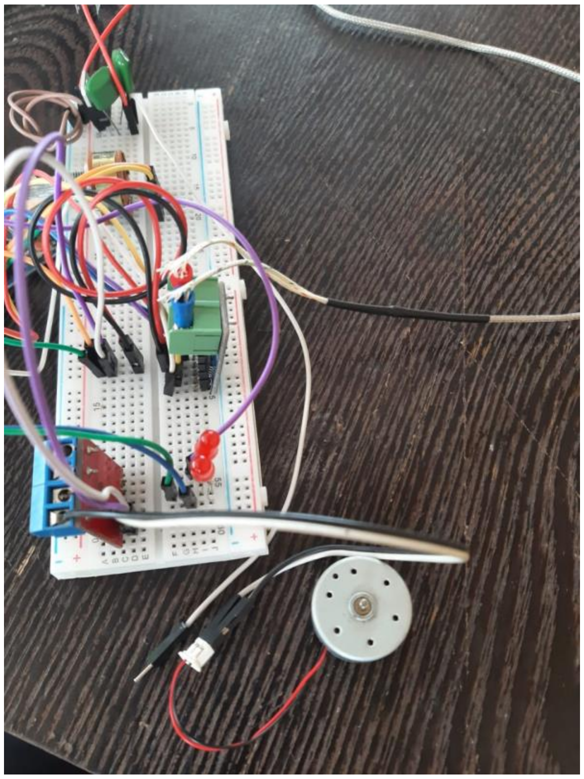


Figure 4.2: Testing using LED's

After this proved to be successful, we next added the DC fan to the circuit as well.

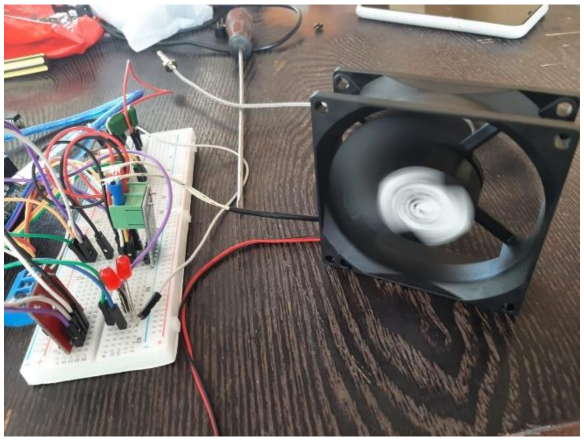


Figure 4.3: Testing the fan

Next, we added the buzzer which would go off after a lapse of one minute given that the temperature isn't within the specified range.

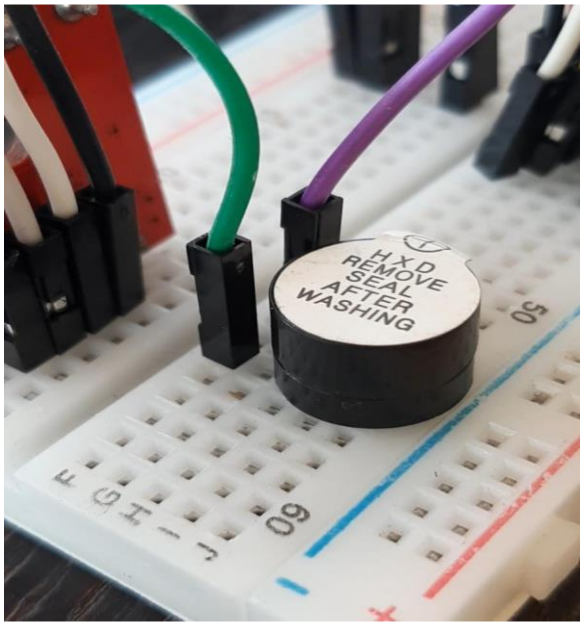


Figure 4.4: Testing the Buzzer

Finally, the OLED was tested to see whether it would give the correct output.

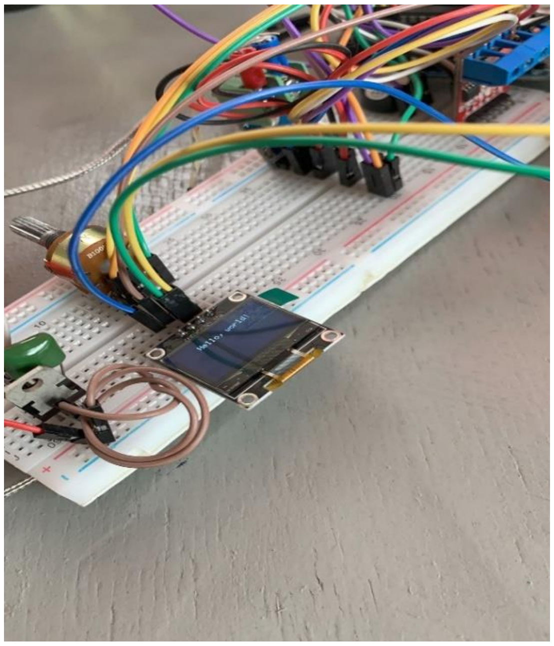


Figure 4.5: Testing the OLED

Breadboard Implementation with the Microcontroller

After the successful implementation of the circuit using an Arduino, next, the circuit was implemented using the standalone ATmega328-PU microcontroller.

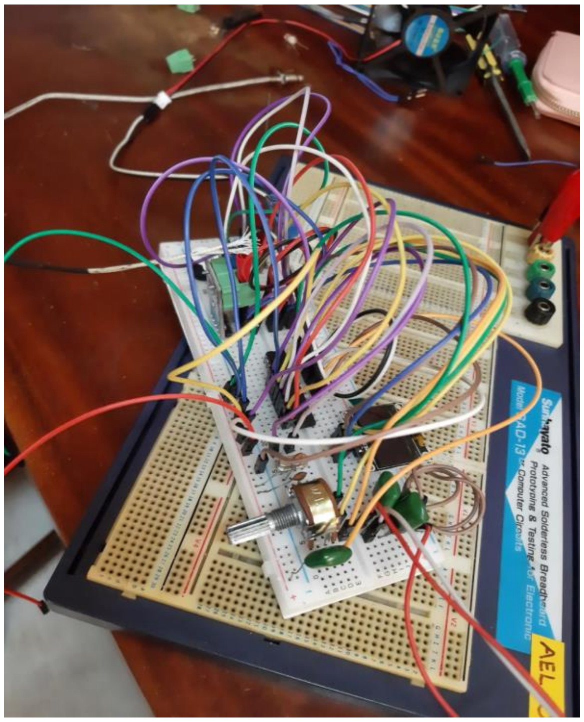


Figure 4.6: Standalone microcontroller testing

PCB Design

PCB design was done using Altium software. In the schematic design, all the components and connections were done according to our circuit design and Arduino implementation. The design was a single layer PCB.

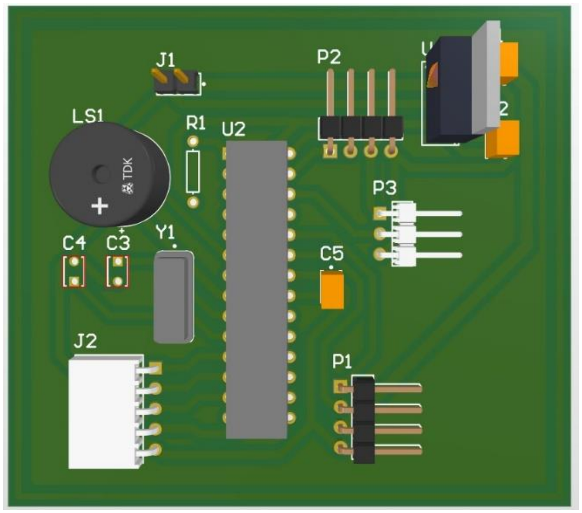


Figure 5.1: 3D view of the PCB

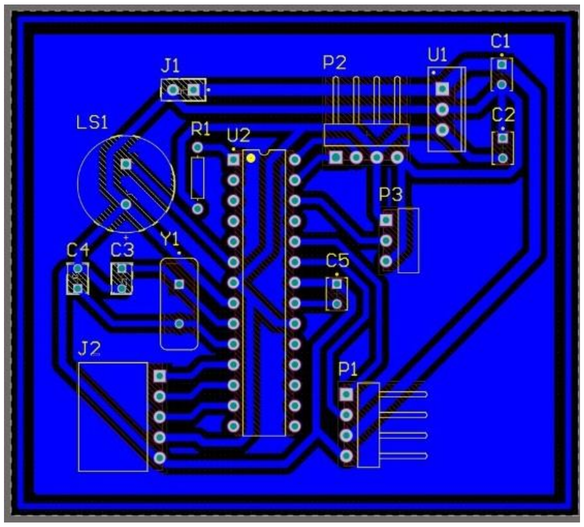


Figure 5.2: PCB design

For the 12V power supply 1.5mm trace width was selected and for the 5V power supply 1mm trace width was selected. The ground had a 1.5mm trace width and the signal lines had a 0.8mm trace width.

These measurements were made taking into consideration the power and current consumption of the circuit.

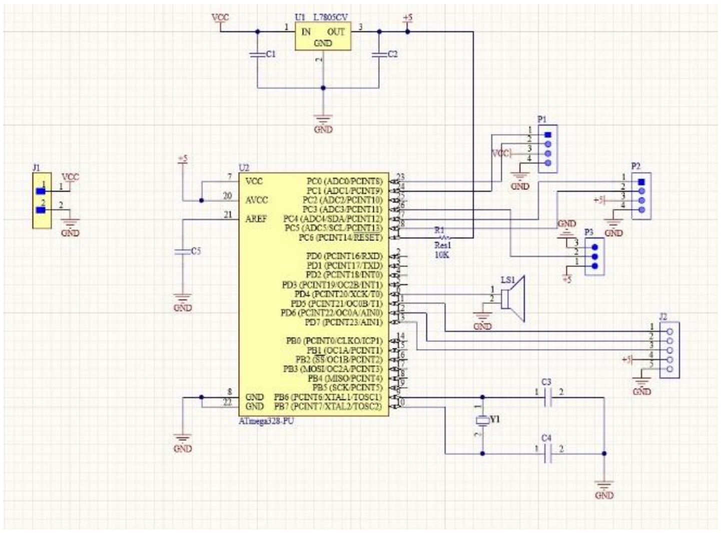


Figure 5.3: Schematic

Enclosure Design

Considering the existing models in the industry, we made an initial sketch on how we wanted the hearth to look like.

To ensure that the cooking pots fit perfectly and to ensure efficient heat transfer, the opening of the hearth was decided to be conical in shape.

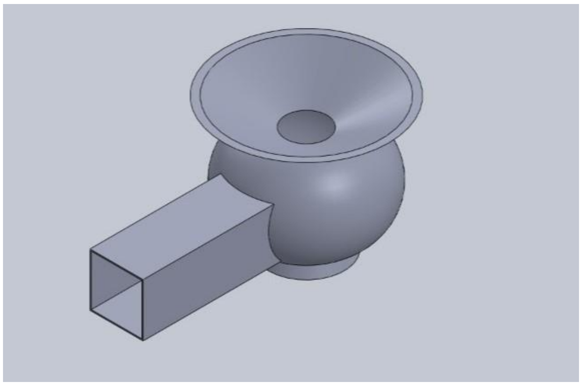


Figure 6.1: Hearth Enclosure

A spherical structure was selected for the base of the hearth as it would be a stronger structure than a cylindrical base.

It was decided that this structure should be made using aluminium as it would pose less of a risk than normal clay pots which are susceptible to cracking. Since aluminium tends to heat up and could be harmful to the user, it was decided that a wooden box should surround the aluminium structure.

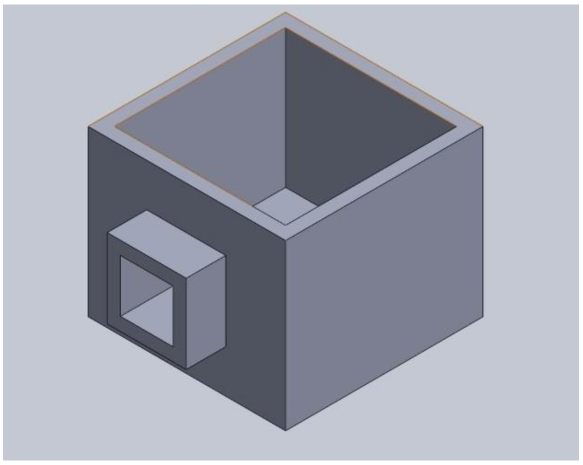


Figure 6.2: Wooden Enclosure

An opening was also made on the Aluminium structure to which the fan was attached. This structure had an opening at the end so that the fan had sufficient access to the atmosphere.

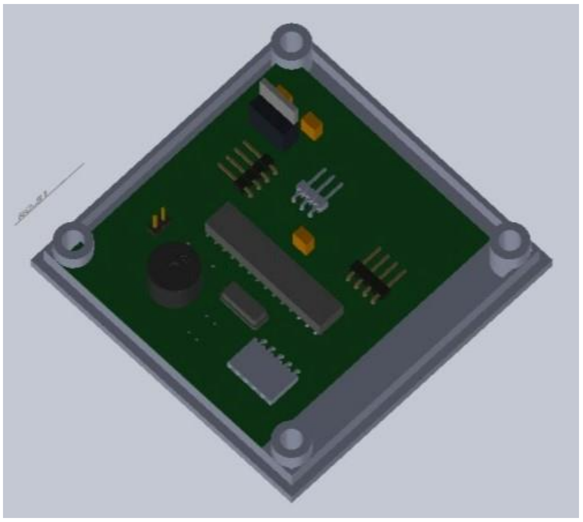


Figure 6.3: PCB Enclosure

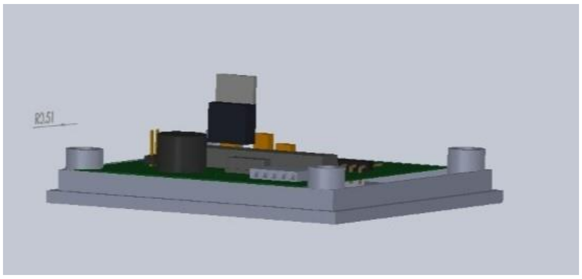


Figure 6.4

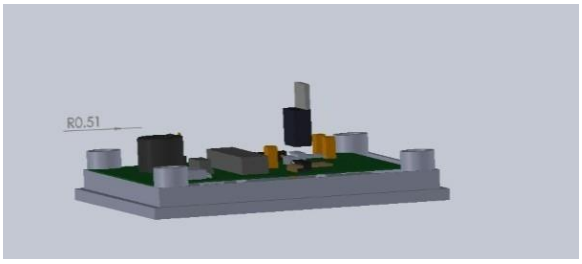


Figure 6.5

On top of the fan enclosure the PCB enclosure was attached, and this enclosure was made taking into consideration the PCB design.

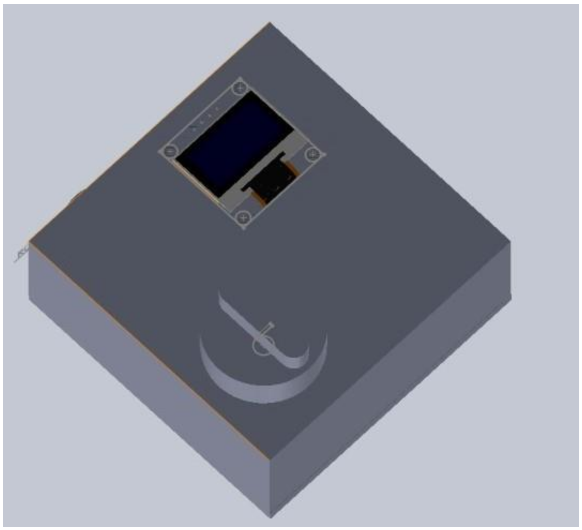


Figure 6.6: Top view of the PCB enclosure

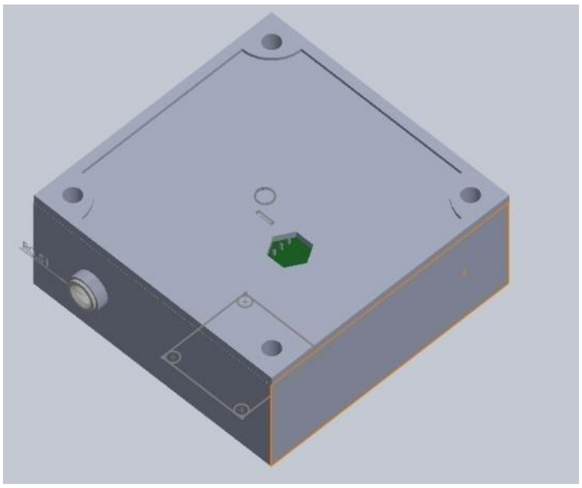


Figure 6.7: Bottom view of the PCB enclosure

For the PCB enclosure, 2 openings had to be made on top to accommodate the LCD as well as the potentiometer by which the user can choose the temperature range. Another opening was required at the bottom of the enclosure to connect the fan as well as to accommodate the temperature sensor.

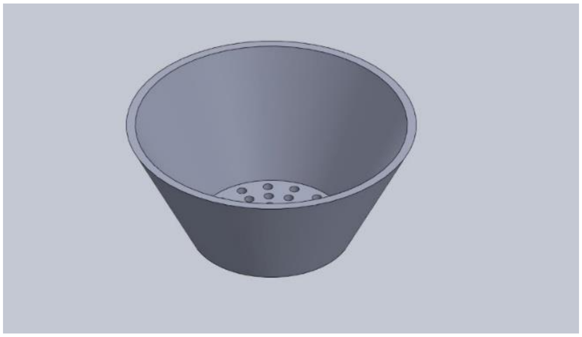


Figure 6.8: Removable opening in the enclosure

One user requirement was to make the removal of coconut husks easy. Therefore, another enclosure was placed at the opening to easily remove and replace and holes at the bottom to ensure that wind still passed through. To protect this enclosure from coconut husks, a mesh was attached to the enclosure to avoid direct contact of coconut husks with the enclosure.

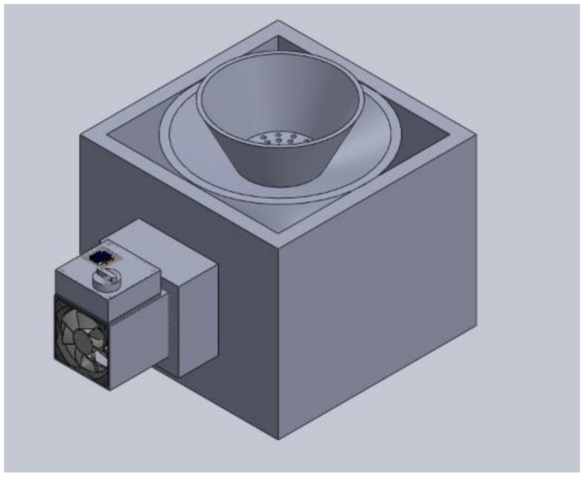


Figure 6.9: Complete enclosure

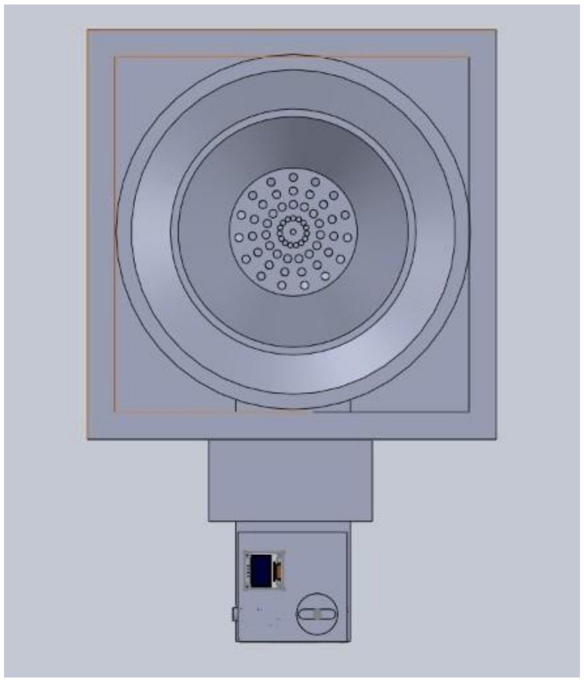


Figure 6.10: Top view of the enclosure

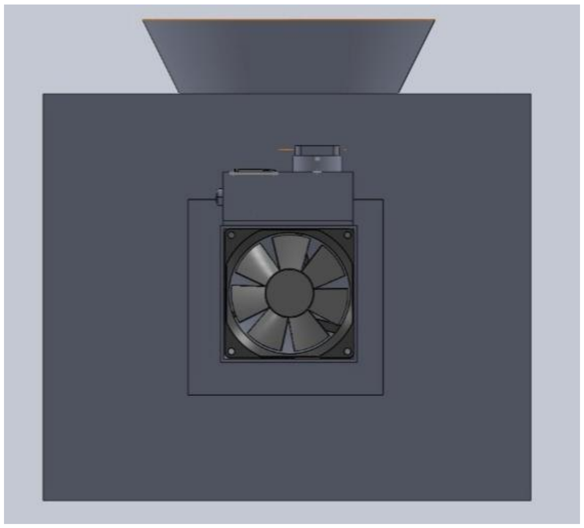


Figure 6.11: Side view of the enclosure

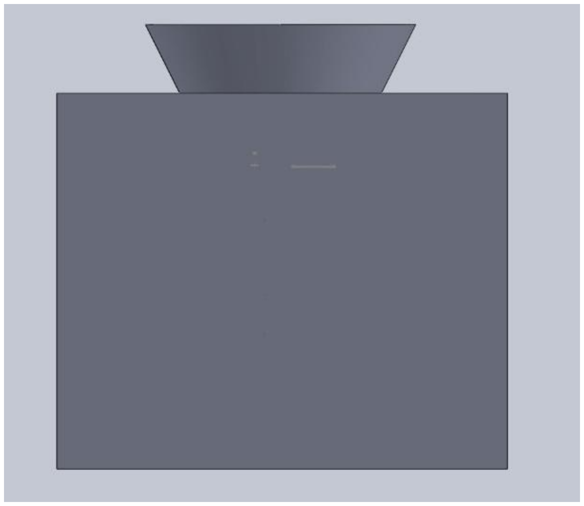


Figure 6.12: Other side view of the enclosure

List of Components

Component	Quantity
ATmega328-PU	1
L7805CV	1
100 nf capacitor	3
22 pf capacitor	2
16 MHz crystal oscillator	1
Piezoelectric buzzer	1
10K resistor	1
2 pin headers	1
4 pin headers	2
3 pin headers	2
5 pin headers	1
230V-12V Power regulator	1
MAX6675 Temperature sensor	1
L9110 Motor module	1
12V DC fan	1
OLED	1
100K Potentiometer	1

ATmega328-PU

The high-performance Pico Power 8bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purposes I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. This is the main component of the microcontroller circuit. Program is coded and fetched into this.

MAX6675 Temperature Sensor

The MAX6675 performs cold-junction compensation and digitizes the signal from a type-K thermocouple. The data is output in a 12-bit resolution, SPI-compatible, read-only format. This converter resolves temperatures to 0.25°C, allows readings as high as +1024°C, and exhibits thermocouple accuracy of 8 LSBs for temperatures ranging from 0°C to +700°C. The MAX6675 is available in a small, 8-pin SO package.

Figure 7.1: Temperature Sensor

Considering the components used, the existing products in the market as well as the enclosure, we determined that our product lifetime is 3 years.

L9110 Motor Driver Module

L9110 Module can control up to 2 DC motors with directional and speed control. Previously we had selected the L298 Motor Driver Module however it has a higher power consumption. Therefore, we chose to switch to this motor driver module instead.

This is used to control the speed of the 12V DC fan.

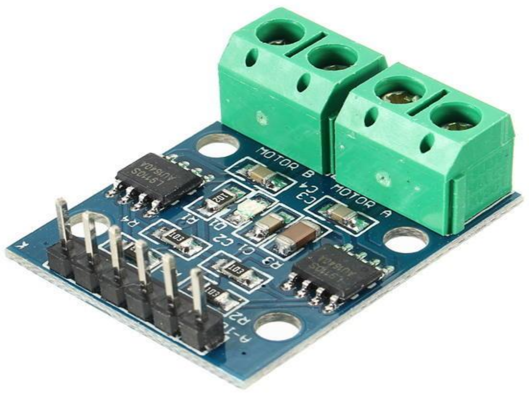


Figure 7.2: Motor Driver Module

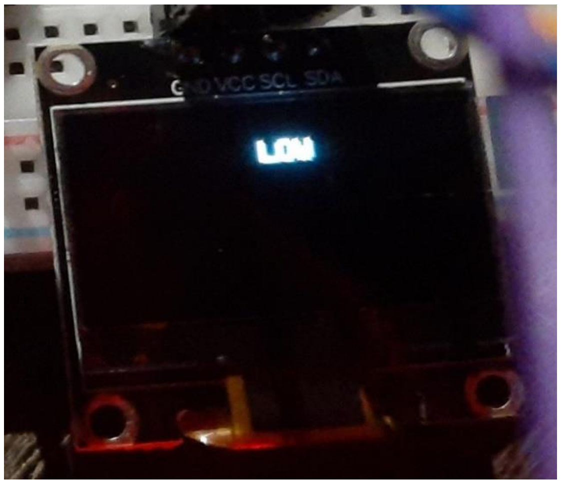
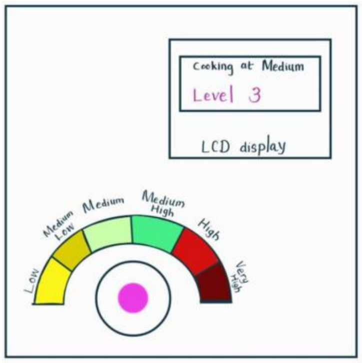


Figure 8.1: Low temperature range.

UI and UX



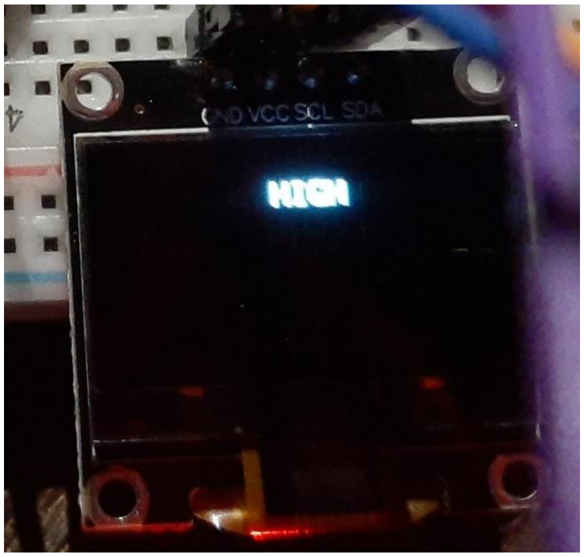


Figure 8.2: High Temperature range

The user interface consists of 2 main components. The OLED and the potentiometer. The potentiometer will have a colour range depicting the 6 different temperature ranges of the device being,

1. Low
2. Medium Low
3. Medium
4. Medium High
5. High
6. Very High

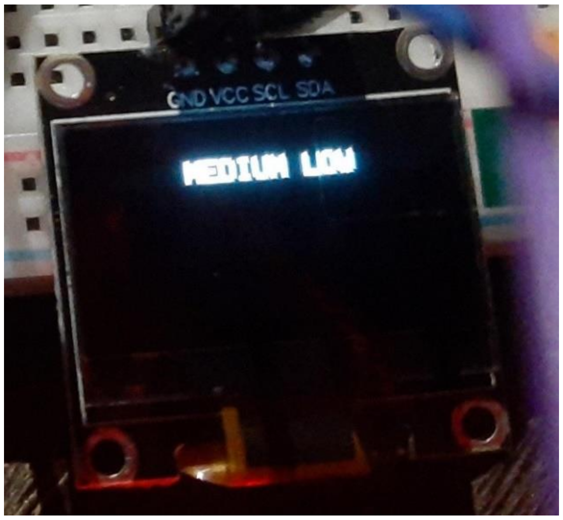


Figure 8.3: Medium Low temperature range

Once one of these ranges are selected by the potentiometer, the output will be shown on the OLED.

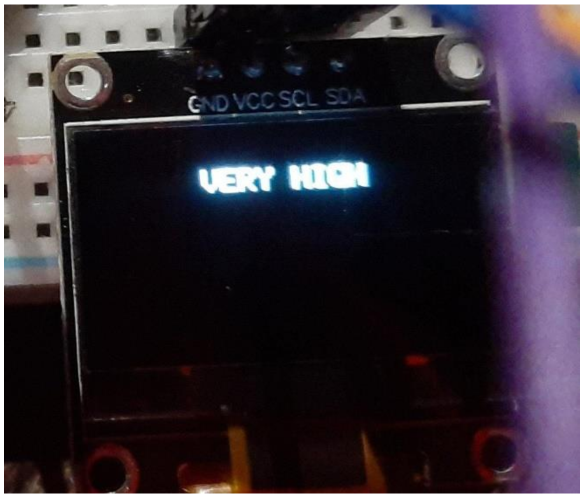


Figure 8.4: Very High temperature range

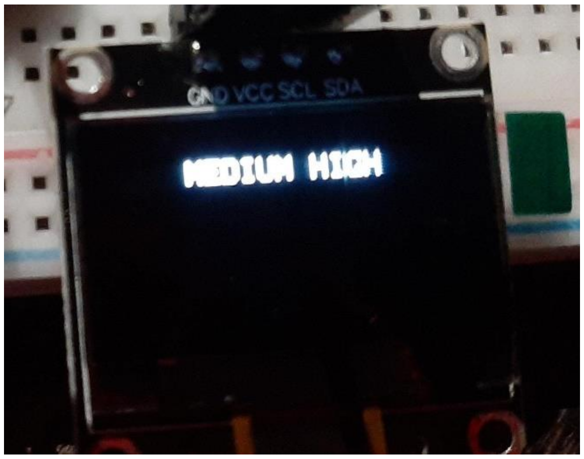


Figure 8.5: Medium High temperature range

Testing

To ensure the functionality of the system, several tests were conducted both for individual components as well as the entire system. The tests conducted are as follows,

- Individual testing of the component functionality using Arduino (mentioned above).
- Testing the heat tolerance of the hearth enclosure (Mentioned below).
- Testing the fan speeds for different temperature ranges by hardcoding input temperatures from the sensor.
- Testing buzzer functionality by hard coding erroneous values.
- Checking different temperature ranges for different potentiometer inputs. This was also used to check whether the OLED was displaying the correct outputs.
- Testing whether the temperature sensor was giving correct outputs by using 2 different temperature sensors.

Results

When the device is initially switched on, the LCD will display "Hello. Choose a Temperature Range." Then the user will have to turn the knob to one of the 6 temperature ranges specified on the enclosure.

Once a temperature range is selected, the LCD will display the selected temperature range. Meanwhile, the temperature sensor will read the current temperature of the coconut husks and will switch on the DC fan to a specified speed to bring the temperature to the selected level.

If the temperature is too high, the fan will switch off and the temperature will drop and once it drops past the selected temperature level, the fan will restart.

In the case that there are some unforeseen circumstances, and the temperature remains too high despite the fan being turned off, the buzzer will go off after a lapse of 1 minute. Similarly, if the temperature doesn't increase despite the fan being turned on, the buzzer will go off again after a lapse of one minute.

Further Improvements

The device can be further improved to accommodate other user requirements such as,

- A system to effectively dispose of the smoke accumulated from the hearth.
- Creating an app so that the user can complete the same tasks through the app and get notifications for abnormal functionality.

Marketing and Sales

Product Packaging

Since the product is not fragile nor does it have any technical specifications for storage, packaging can be done using a normal cardboard box with bubble wrap. A leaflet will also be given with instructions on how to use the hearth.

Maintenance and repair

The warranty period for the hearth will be a period of 3 years and for any issues after sale services will be provided through either repair of the existing hearth or replacing it with a new hearth.

Disposal

Some components, such as the aluminium can be reused or recycled at the end of the product lifetime and therefore ways in which it can be reused will be mentioned on the leaflet. However, other electronic components will have to be disposed of in a safe manner and instructions for disposal will be given to the user in the leaflet.

Discussions

Determining the material of the hearth.

A main concern when determining the material of the hearth was the melting point or cracking point of the material. Using clay for the pot would mean that the pot would crack at our maximum temperatures but there was also a concern as to whether aluminium would melt at our highest temperature range. Therefore, the material was tested by holding it to a high flame and observing its' results. No negative results were observed.

OLED display terms were overlapping.

When changing the temperature range from one user input to another, the display would overlap the terms for a period of 2 seconds before completely changing. This was because the outputs were of different sizes and therefore for a complete rewrite of the output, padding needed to be added so that all outputs were of the same size.

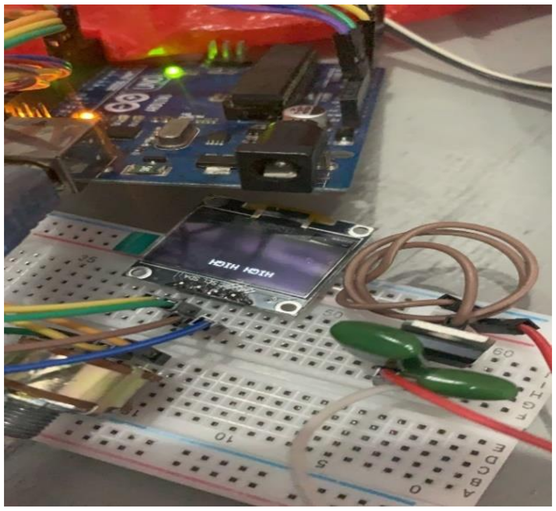


Figure 10.1: An overlap of "VERY HIGH" with "HIGH"

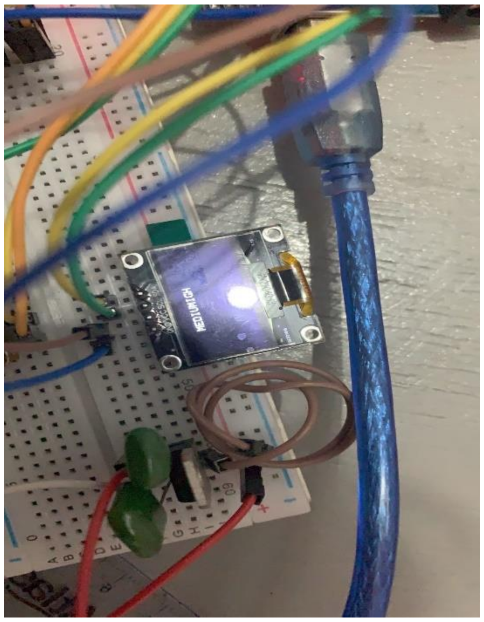


Figure 10.2: An overlap of "MEDIUM" with "HIGH"

Determining the count for the buzzer.

The buzzer is said to go off at the time of an issue. But we needed to determine the time needed for the buzzer to go off. Our code gets temperature readings every one second. To move the temperature from one range to another it would take some time and the time taken would depend on many factors. Through testing we determined that the best possible time lapse to wait until the buzzer goes off is 10 minutes as some temperature range shifts would take this amount of time whereas if there was an issue, a 10-minute lapse won't be too disastrous.

Acknowledgement

We would like to extend our gratitude to our lecturers who provided us with the required technical knowledge and guidance which served as the foundation for this project.

Bibliography

Guide for I2C OLED Display with Arduino. Retrieved from

https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/

Spherical and Cylindrical Pressure Vessels. Retrieved from

https://www.wermac.org/equipment/pressurevessel.html

Getting Started With the ATMega328P. Retrieved from

https://www.instructables.com/Getting-Started-With-the-ATMega328P/

Temperature based Fan Speed Control. Retrieved from https://create.arduino.cc/projecthub/embeddedlab786/temperature-based-fan-speed-control-945f9d

WHEN TO USE ALUMINUM VS STAINLESS STEEL. Retrieved from

https://www.kloecknermetals.com/blog/when-to-use-aluminum-vs-stainless-steel/#:~:text=When%20comparing%20stainless%20steel%20vs,over%20steel%20in%20cold%20temperatures.

TODD OPPENHEIMER. The Laws of Thermo-Culinary Dynamics. Retrieved from

https://craftsmanship.net/sidebar/the-laws-of-thermo-culinary-dynamics/#:~:text=Aluminum%20is%20even%20more%20aggressive,than%20its%20equivalent%20in%20clay.

Maureen. Induction Cooking Temperature Guide With Settings & Chart. Retrieved from

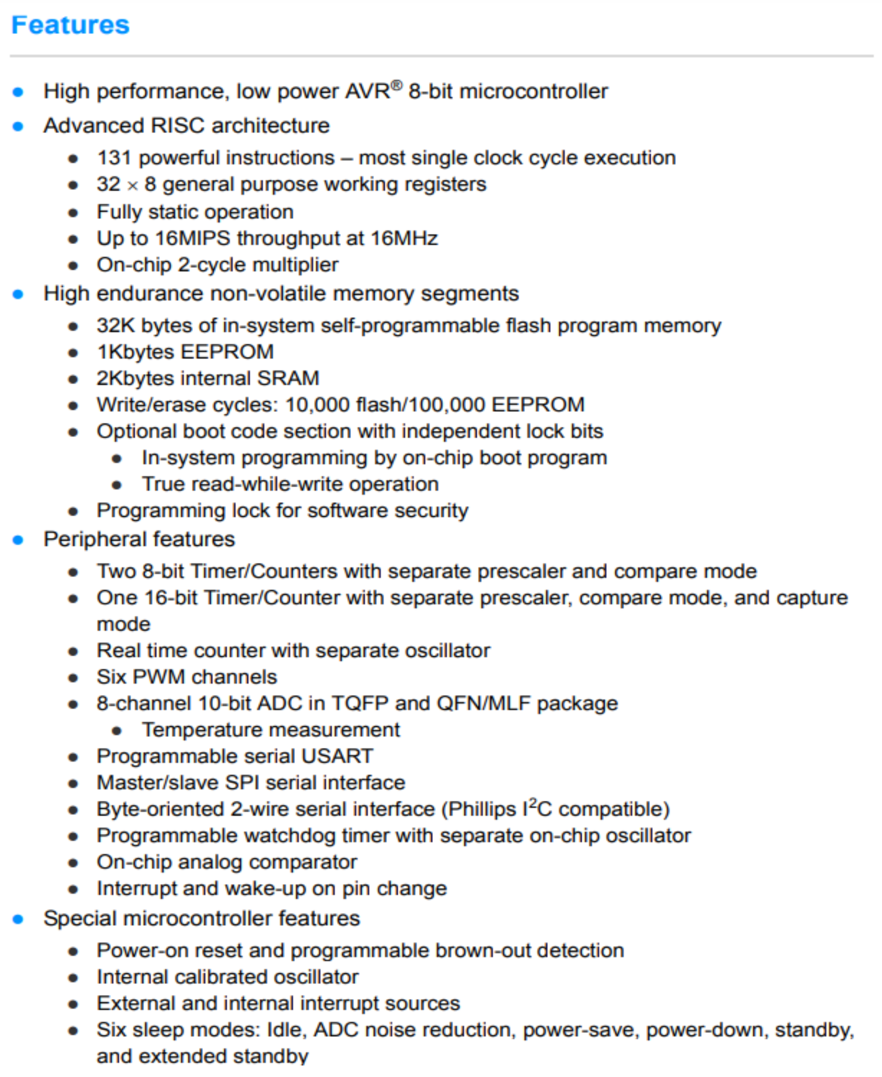
https://apreparedkitchen.com/induction-cooking-temperature-settings/

Appendices

Budget

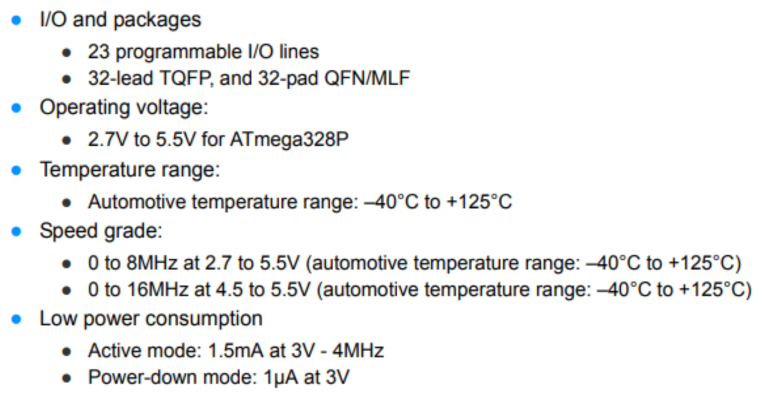
Component	Quantity	Cost
ATmega328-PU	1	1700
L7805CV	1	60
100 nf capacitor	3	15
22 pf capacitor	2	20
16 MHz crystal oscillator	1	40
Piezoelectric buzzer	1	240
10K resistor	1	45
2 pin headers	1	90
4 pin headers	2	
3 pin headers	2	
5 pin headers	1	
230V-12V Power regulator	1	1380
MAX6675 Temperature sensor	1	1350
L9110 Motor module	1	200
12V DC fan	1	620
OLED	1	1000
100K Potentiometer	1	60
PCB		550
Enclosure		2000
Total		9370

ATmega328P Datasheet Summary



Features

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^2C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby



- I/O and packages
  - 23 programmable I/O lines
  - 32-lead TQFP, and 32-pad QFN/MLF
- Operating voltage:
  - 2.7V to 5.5V for ATmega328P
- Temperature range:
  - Automotive temperature range: –40°C to +125°C
- Speed grade:
  - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
  - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
  - Active mode: 1.5mA at 3V - 4MHz
  - Power-down mode: 1µA at 3V

MAX6675 Temperature Sensor Datasheet Summary

General Description

The MAX6675 performs cold-junction compensation and digitizes the signal from a type-K thermocouple. The data is output in a 12-bit resolution, SPI-compatible, read-only format.

This converter resolves temperatures to 0.25°C, allows readings as high as +1024°C, and exhibits thermocouple accuracy of 8 LSBs for temperatures ranging from 0°C to +700°C.

The MAX6675 is available in a small, 8-pin SO package.

Applications

- Industrial
- Appliances
- HVAC

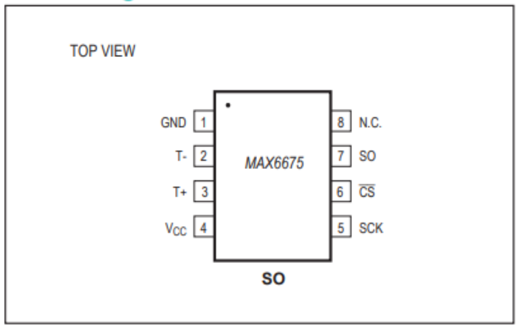
Features

- Direct Digital Conversion of Type -K Thermocouple Output
- Cold-Junction Compensation
- Simple SPI-Compatible Serial Interface
- 12-Bit, 0.25°C Resolution
- Open Thermocouple Detection

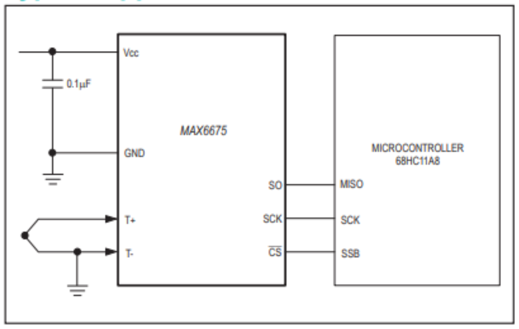
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX6675ISA	-20°C to +85°C	8 SO

Pin Configuration



Typical Application Circuit



Absolute Maximum Ratings

Supply Voltage (V_{CC} to GND)	-0.3V to +6V
SO, SCK, $\overline{CS}$, T-, T+ to GND	-0.3V to V_{CC} + 0.3V
SO Current	50mA
ESD Protection (Human Body Model)	±2000V
Continuous Power Dissipation (T_A = +70°C)	
8-Pin SO (derate 5.88mW/°C above +70°C)	471mW
Operating Temperature Range	-20°C to +85°C
Storage Temperature Range	-65°C to +150°C
Junction Temperature	+150°C
SO Package	
Vapor Phase (60s)	+215°C
Infrared (15s)	+220°C
Lead Temperature (soldering, 10s)	+300°C

Code

```
#include "max6675.h" // max6675.h file is part of the library that you should download from Robojax.com
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

int soPin = 5;// SO=Serial Out
int csPin = 6;// CS = chip select CS pin
int sckPin = 7;// SCK = Serial Clock pin
int analogPin = A3;
int celTemp;
int tempSet;
int fanPin1 = A1;
int fanPin2 = A0;
int T1;
int T2;
MAX6675 robojax(sckPin, csPin, soPin);// create instance object of MAX6675
bool prevLower = false; // keep track of whether temperature was outside range in previous iteration of loop
bool prevHigher = false;
int prevCount = 0; // keep track of iterations where temperature was below or above range consecutively
String prev_disp;

void setup() {

  pinMode(analogPin, INPUT);
  pinMode(fanPin1, OUTPUT);

  pinMode(fanPin2, OUTPUT);
  Serial.begin(9600);// initialize serial monitor with 9600 baud
  Serial.println("Robojax MAX6675");

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(1.5);
  display.setTextColor(WHITE);
  display.setCursor(30, 10);
  display.println("   HELLO   ");
  display.display();
  prev_disp = "   HELLO   ";
}

void disp(String txt1,String txt2){  //OLED Display Function
  if(txt1!=txt2){
    display.setCursor(30, 10);
    display.setTextColor(BLACK);
    display.println(txt2);
    display.display();
    display.setCursor(30,10);
    display.setTextColor(WHITE);
    display.println(txt1);
    display.display();
  }
}
```

```
void loop() {
  // basic readout test, just print the current temp
  // Robojax.com MAX6675 Temperature reading on Serial monitor
   celTemp=robojax.readFarenheit();
   tempSet=analogRead(analogPin);
   Serial.print("F = ");
   Serial.print(celTemp);
   //Serial.print(" F = ");
   //Serial.println(robojax.readFahrenheit());
   Serial.print(" Controller= ");
   Serial.print(tempSet);

 //celTemp = 200;

   if(tempSet<200){
    T1 = 50;
    T2 = 120;
    disp("     LOW    ",prev_disp);
    prev_disp = "     LOW    ";
   }else if(tempSet<300){
    T1 = 120;
    T2 = 150;
    disp(" MEDIUM LOW",prev_disp);
    prev_disp = " MEDIUM LOW";
   }else if(tempSet<400){
    T1 = 150;
    T2 = 190;
    disp("  MEDIUM   ",prev_disp);
    prev_disp = "  MEDIUM   ";
   }else if(tempSet<500){

    T1 = 190;
    T2 = 231;
    disp("MEDIUM HIGH",prev_disp);
    prev_disp = "MEDIUM HIGH";
   }else if(tempSet<600){
    T1 = 231;
    T2 = 288;
    disp("    HIGH    ",prev_disp);
    prev_disp = "    HIGH    ";
   }else if(tempSet<700){
    T1 = 288;
    T2 = 372;
    disp(" VERY HIGH ",prev_disp);
    prev_disp = " VERY HIGH ";
   }

   if(celTemp<T1){
    analogWrite(fanPin1,0);
    analogWrite(fanPin2,255);
    if (prevLower) {
         prevCount++;
      } else {
        prevLower = true;
        prevHigher = false;
        prevCount = 1;
      }
   }else if(celTemp<T2){
    analogWrite(fanPin1,0);
    analogWrite(fanPin2,140);
    prevCount = 0;
    prevLower = false;
```

```
      prevHigher = false;
     }else{
      analogWrite(fanPin1,0);
      analogWrite(fanPin2,0);
      if (prevHigher) {
           prevCount++;
        } else {
          prevLower = false;
          prevHigher = true;
          prevCount = 1;
        }
     }

     if (prevCount > 600) { // if temperature has consecutively been lower or higher for n number of iterations then do something

      tone(4,255);
      display.setTextColor(WHITE);
      display.setCursor(30, 20);
      display.println("   ERROR   ");
      display.display();
      }
    else{
      display.setTextColor(BLACK);
      display.setCursor(30, 20);
      display.println("   ERROR   ");
      display.display();
      noTone(4);
      }

    Serial.print("\n loop");
    Serial.print(prevCount);

    Serial.print(prevLower);
    Serial.print(prevHigher);

     delay(1000);
  }
```