Faculty of Information Technology

Kithul Treacle and Jaggery Maker

Final Report

GROUP 18

Index No:	Name
205050U	**KAPUWATHTHA H.B.M.R.M.M**
205093D	SALAMA F.H. F
205108G	THENNAKOON T.M.M. S
205096N	SAMARAKKODI S.A.T. S

Supervisor Name: Mr.BH Sudantha,
Senior Lecturer,
Dept. of Information Technology

Co-Supervisor Name: Miss. Chamalka Rajapaksa
Lecturer
Dept. Of Information Technology

Date of Submission: 06/06/2022

Table of Content

1.0 Introduction

The Kithul Treacle and Kithul Jaggery considered to be one of the most important traditional sweetener types. The sap of the Kithul palm is used to produce Kithul Treacle and Kithul Jaggery, which have high nutrition.

There is a high demand for Kithul Treacle and Kithul jaggery in our country as well as in the international market. The making of Kithul products in Sri Lanka has been followed traditionally in some regions. But today the demand for Kithul products is far more than the production and supply.

But still, there is no technical and automated solution or method for Kithul Treacle and Jaggery production. So, we are going to design a Kithul Treacle and Kithul Jaggery making machine for domestic usage and for the owners of small-scale industries targeting the local market. The innovative system that we propose offers a unique and automated way to produce Kithul Treacle and Jaggery easily. The specialty of this machine is, we can make Kithul Treacle and Jaggery in an automated way and experience or much knowledge about Kithul making is not required.

As we are living in an era where technology has already reached every corner in Sri Lanka, this will be a very useful and effective innovation for Kithul Treacle and Kithul Jaggery production.

2.0 Problem in Brief

People must face a lot of difficulties and issues due to the traditional way of making Kithul treacle and jaggery.

- Using a traditional method to produce Kithul Treacle and Kithul Jaggery takes more time and effort.
- When we make the Kithul Treacle and Kithul Jaggery, continuous monitoring and heat controlling are required.
- The sap of the Kithul flower mixture should be mixed continuously, until the end of the process.
- To produce Kithul Treacle and Kithul Jaggery, we have to use wood fire. It is not easy to maintain the wood fire at the same level till the end.
- When the person who prepares Kithul Treacle and Kithul Jaggery has to stay near the wood fire to monitor the process, there is a high chance to get skin burns.
- Producer should have a proper understanding of the thickness level for making treacle and jaggery.

3.0 Literature Survey

Following are some projects we have identified slightly similar to our project,

3.1 Kithul jaggery cube making machine

This product is used to make the jaggery cube using brown sugar, white sugar or jaggery syrup. Here the jaggery mixture is prepared using the traditional hand mixing method without any technical means. It is a good processing line with low cost.

The processing line will automatically and continuously fill the jaggery sugar syrup into the mould. The process is like this. First, pour the finished jaggery sugar syrup into the storage hopper with heating preservation. Then the filling head will automatically weigh and fill, feeding the jaggery to cooling tunnel to form cool, cooled formed jaggery solid the mould to packing, empty mould go on filling recycle.

Figure *1* Kithul jaggery cube making machine

Figure 2 Kithul jaggery cube making machine

3.2 Jaggery making plant

This product is used to make the jaggery using the jaggery syrup. The specialty of this product is there is an automated mixture part to mix the jaggery syrup. The mechanism which they use to mix the jaggery syrup helps to overcome the limitations of traditional jaggery making methods

Figure 3 Jaggery making plant

Figure 4 Jaggery making plant

4.0 Aim and Objectives

4.1 Aim:

Our aim is to design and develop an automated machine to produce Kithul Treacle and Kithul Jaggery for domestic usage

4.2 Objectives:

- To overcome the limitations of traditional methods.
- To bring an innovative Kithul Treacle and Jaggery making machine to the market.
- To reduce the time and energy wastage of user.
- To meet customer's expectations by providing well performing, easy to operate machine
- To easily operate the Kithul Treacle and Jaggery making machine simply with a keypad.
- To let anyone, make Kithul Treacle and Jaggery at home easily.

5.0 System Description

5.1 Description

Our Proposed Solution is to design a Kithul Treacle and Kithul jaggery-making machine. The innovative system that we propose offers a unique and automated way to make Kithul Treacle and Kithul Jaggery production easy.

Initially, we use a coil to heat the sap of the Kithul flower. As the first step user should pour the sap of the Kithul flower into the filtering part. There is a weight sensor that is fixed to the filter to measure the weight of sap. The maximum weight of sap that can be poured is 4Kg. So, if the weight is greater than 4Kg, the weight will be displayed on the LCD display and the buzzer will start ringing. Indicating that no more sap can be poured. (To make a bottle of Kithul Treacle, we need about 4Kg sap of the Kithul flower.) After measuring the weight and ringing buzzer, the filtering valve will open. Then the sap will be filtered and poured into the container.

Then the user can choose whether he/she wants to produce Kithul Treacle or Kithul Jaggery using keypad which is fixed to the container.

And there is a temperature sensor to measure the temperature of sap continuously until the heating stops.

We have implemented two separated keys to make it easy for the user to tell the machine what should be prepared.

If the user presses the respective key to make Kithul Treacle, the sap will be heated until the temperature increases to103ºC, once the mixture turns to 103ºC, the heating will be stopped. In addition to, if the user wants to heat the sap further, user can enter the temperature through the keypad and start to heat. After the mixture turns to respective temperature the heating stops, then the user can open the Kithul Treacle valve to take out Kithul Treacle from the container by using keypad.

And if the user presses the respective key to make Kithul Jaggery, the sap will be heated until the temperature is equal to 200ºC, once the mixture turns to 200ºC, the heating will be stopped. Here also, if the user wants to heat the sap further, user can heat further. Then user can open the Kithul Jaggery valve to take out Kithul Jaggery from the container by using keypad.

When making Kithul Treacle or Kithul Jaggery manually, user have to mix the mixture continuously. So, we have designed an automated hand to mix the Sap continuously. Once the user presses respective key to make Kithul Treacle or Jaggery the automated hand will start to work.

5.2 Block Diagram

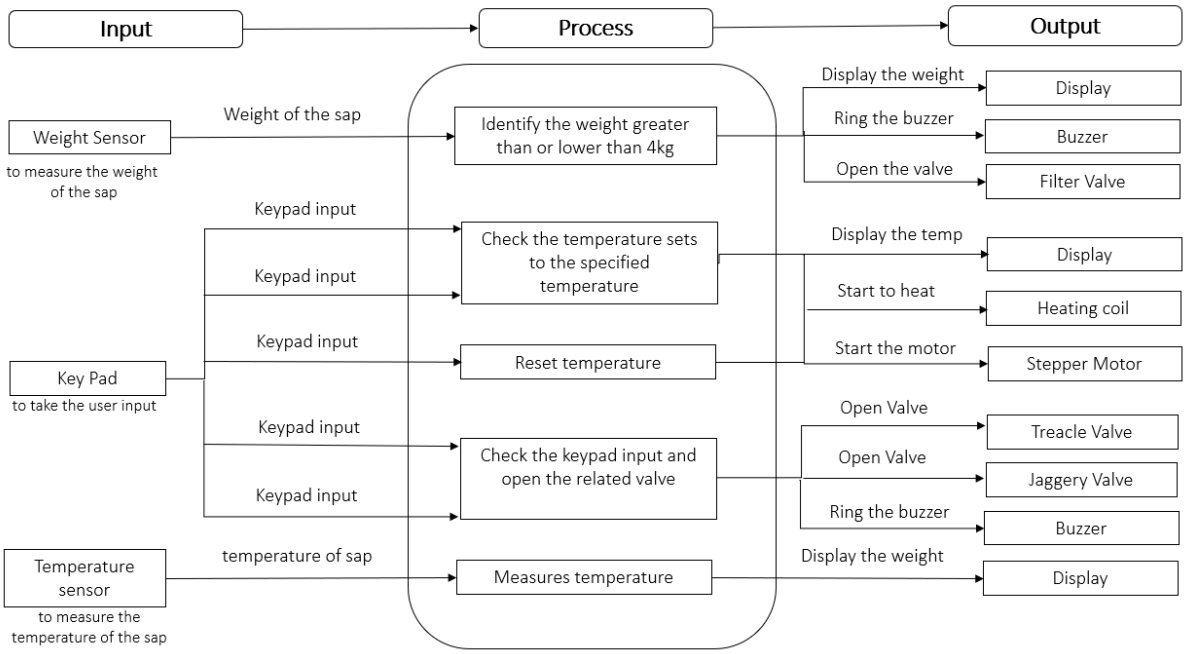


Figure 5 Block Diagram

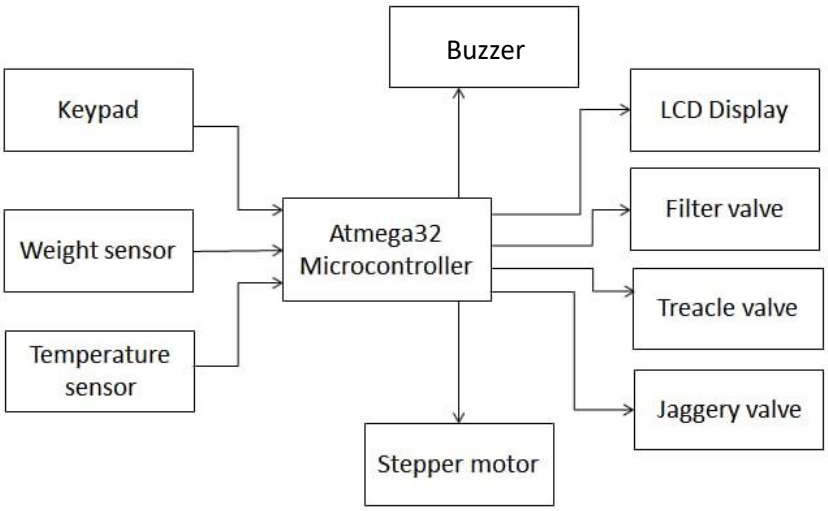


Figure 6 Block Diagram

5.3 Design

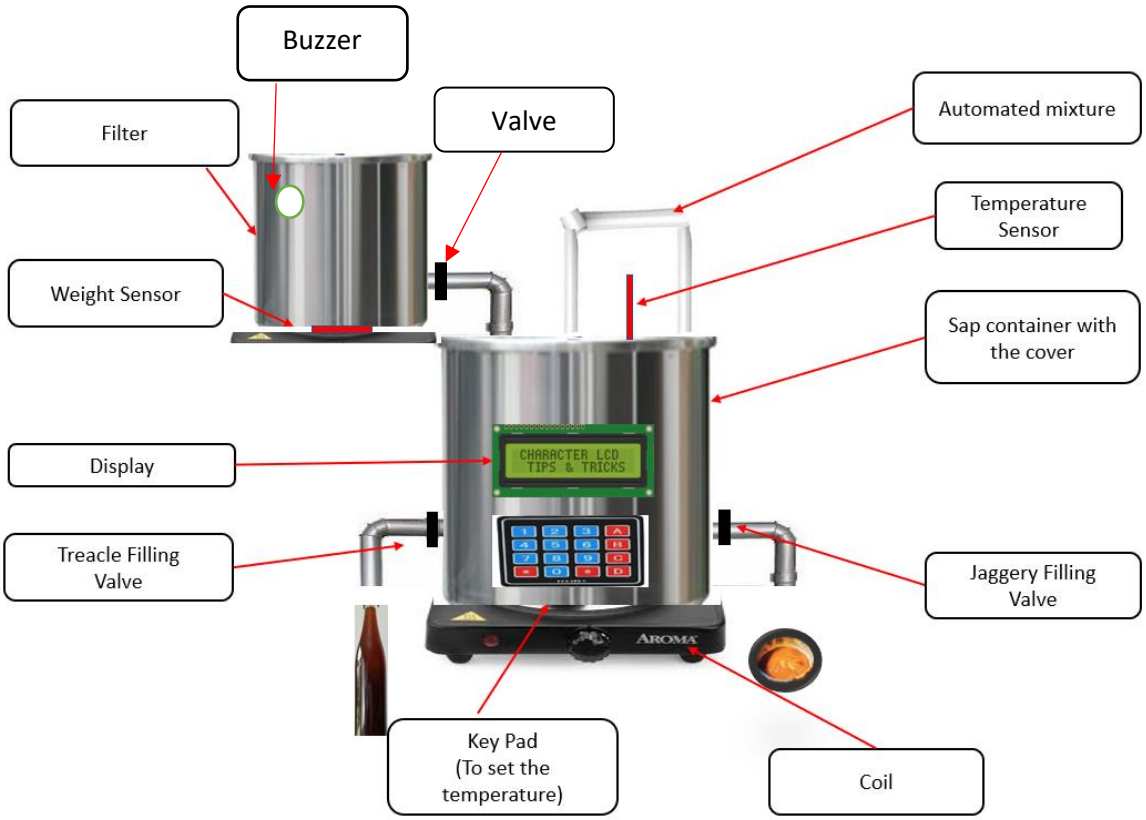


Figure 7 Design

5.4 3D – Design

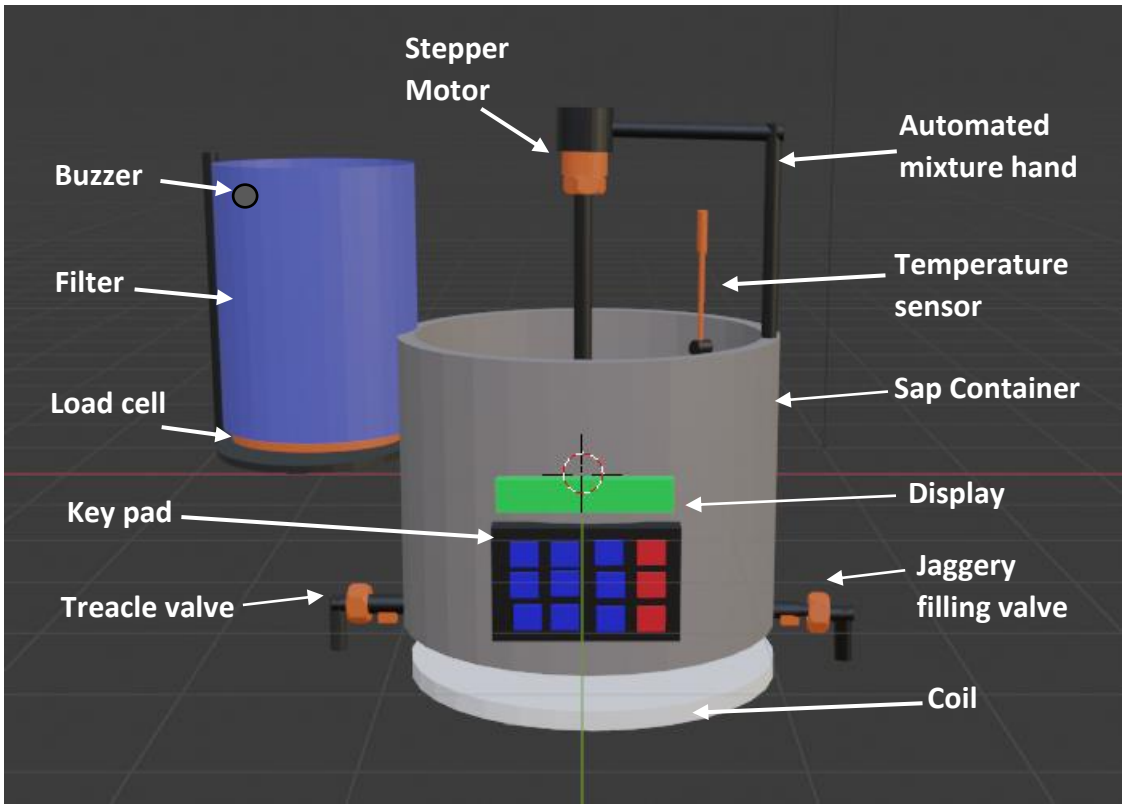


Figure 8 3D Design

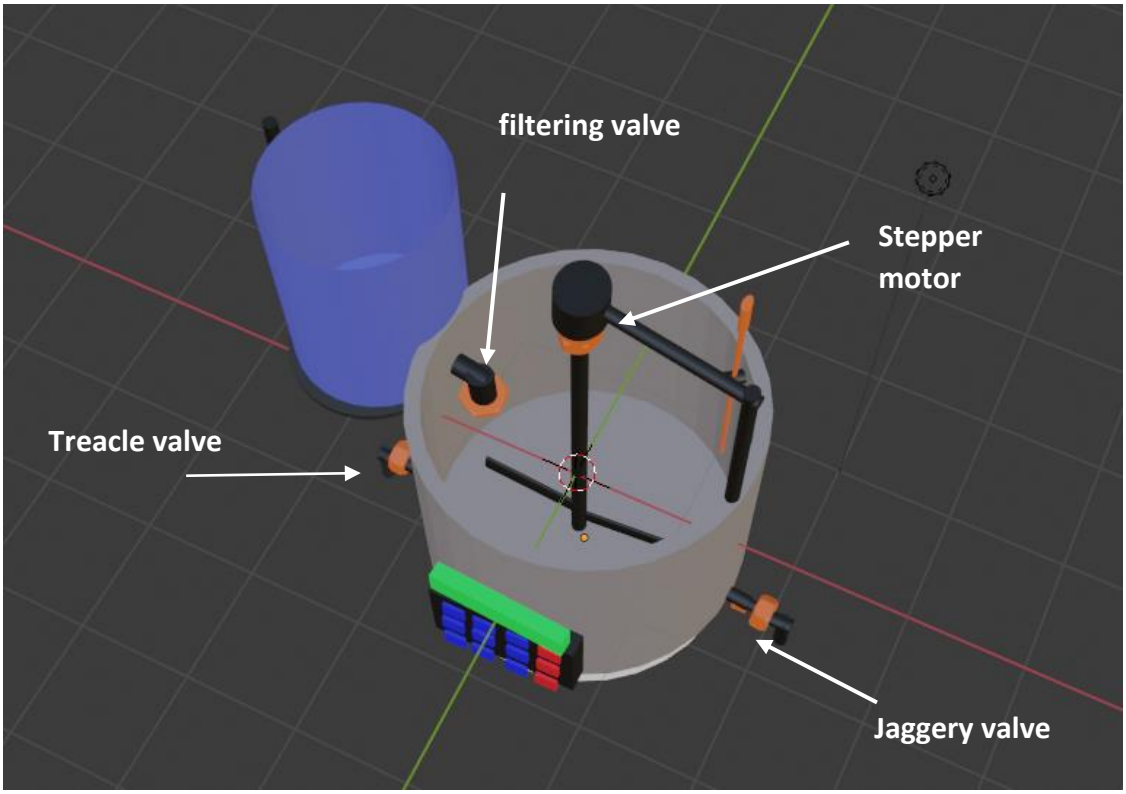


Figure 9 3D Design

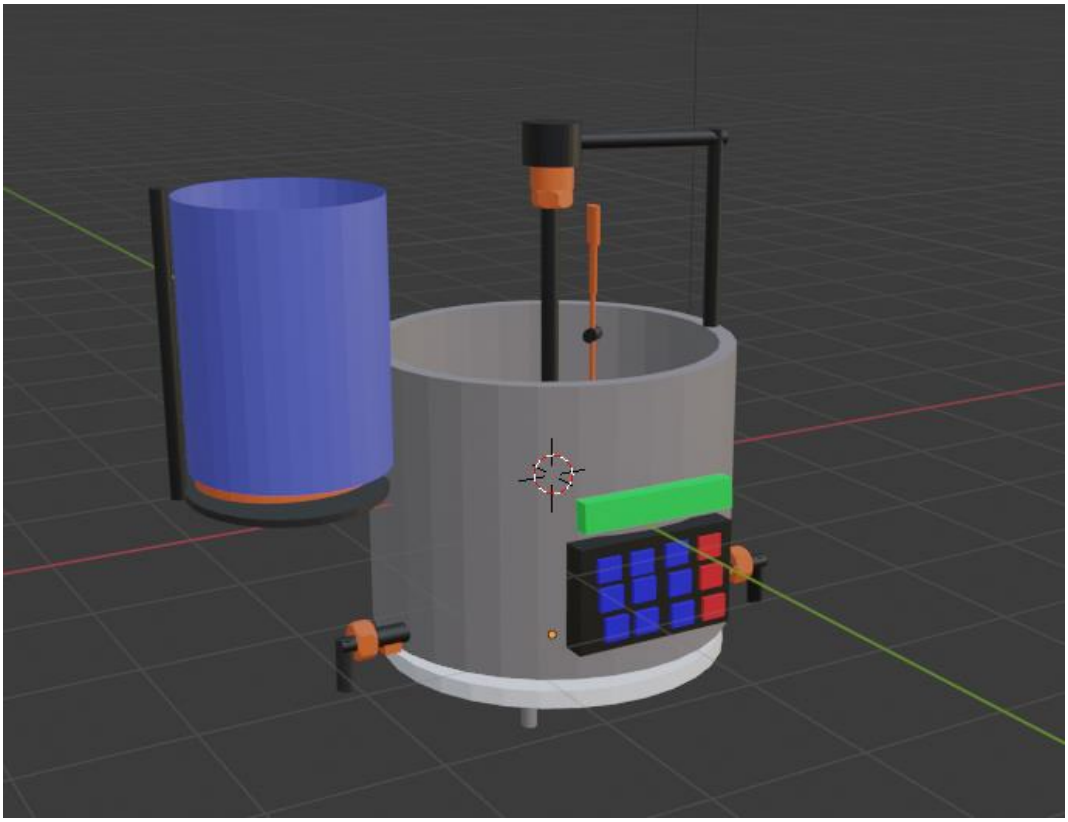


Figure 10 3D Design

6.0 Testing and Implementation

By the Testing and Implementation stage, we have studied and identified what are the components, specifications and techniques, pin diagram, how the components should be connected each other, how to program microcontrollers using Atmel Studio, and how to simulate the simple basic circuits using Proteus. Then we connect the component to atmega32 microcontroller in proteus software and program using Atmel Studio.

All the group members were the part of the whole process. According to the individual responsibilities all the members had done their duties. So, the circuit diagram for the individual components was drawn separately. To design the circuit diagram, we use proteus software. When we designed the circuit diagram we could find there are no some component in proteus software. In our project we have to use solenoid valves. But in proteus software that component was not available. So, we decide to use a relay module to demonstrate the solenoid valve. And for demonstrate the heating coil too we use a relay module. There was not such an issue with the other components. After that, integrate all the components and design the full circuit diagram for entire project.

After completing the circuit design all the group members started to code and complete it. We completed the coded for components separately. We use Atmel Studio software code the project. Here we did the coding part components wise and then complete the entire code. Then the final code was converted to hex file and import to drawn circuit diagram.

We have simulated the full code in proteus simulation and we were able to get the correct simulation due to our hard work and dedication. After that we could design the PCB design for the project and complete the entire project successfully.

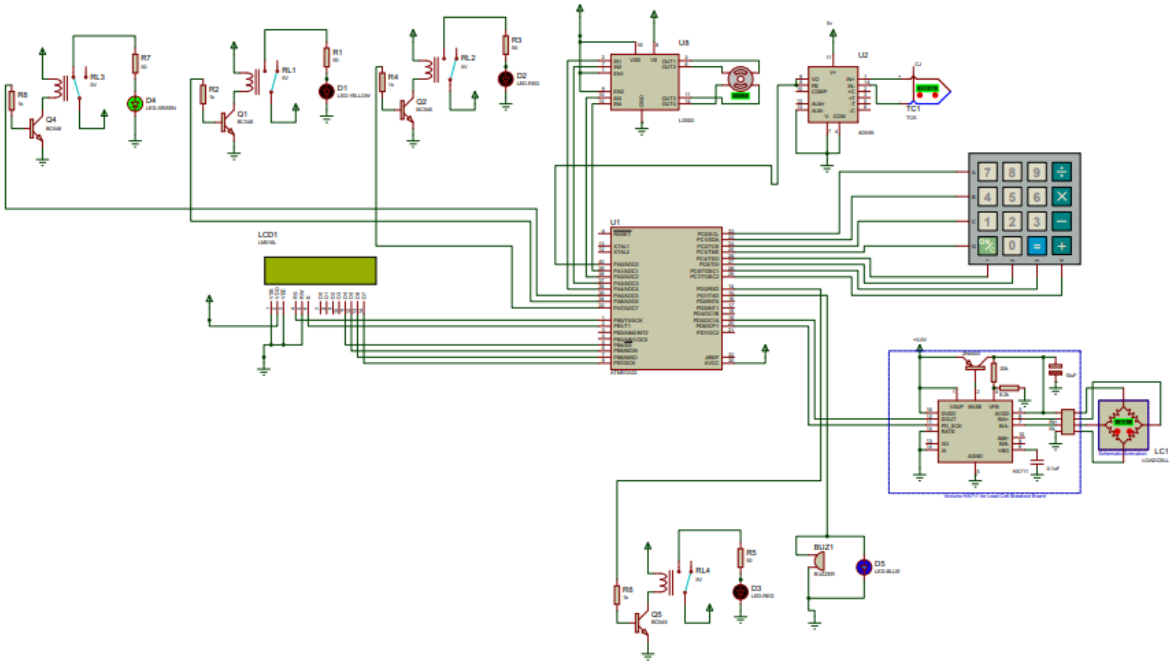


Figure 11 Testing and Implementation

8.0 Resource Requirements

8.1 Hardware Requirements

- Weight Sensor (Straight bar load cell)
- Hx711 load cell Amplifier
- Temperature Sensor (Thermocouple -TC10-0)
- AD595 (Monolithic Thermocouple Amplifiers)
- LCD Display (16*2)
- Buzzer (Piezoelectric Buzzer - PS1240P02BT)
- Solenoid valve (US Series High-Temperature Solenoid Valves)
- Microcontroller (ATMEGA 32)
- Stepper Motor (NEMA 23 bipolar stepper motor)
- L293D Motor Driver
- Keypad (4*4 Keypad)
- Heating coil
- Transformer
- Capacitor
- 7815 power regulators
- Rectifier
- 7812 power regulators
- 7805 power regulators

8.2 Software Requirements

- Atmel Studio
- Blender
- 3D Max
- Proteus
- EasyEDA, Kicad

9.0 Cost Estimate

Name	Quantity	Amount
Straight Bar Load Cell	1	165.00
Hx711 Amplifier	1	200.00
Thermocouple -TC10-0	1	700.00
Monolithic Thermocouple Amplifiers	1	700.00
LCD Display 16*2	1	400.00
Piezoelectric Buzzer	1	75.00
Solenoid Valves	3	1500.00
Atmega 32 Microcontroller	1	1010.00
NEMA 23 bipolar stepper motor	1	1500.00
L293D Driver	1	490.00
Heating coil	1	2500.00
Keypad	1	260.00
Heating Coil	1	2500.00
Transformer	1	1000.00
Cover	1	2500.00
Other	1	2000.00
Total		**16800.00**

10.0 Individual Contribution

10.1 205050U - KAPUWATHTHA H.B.M.R.M.M

Responsible Part

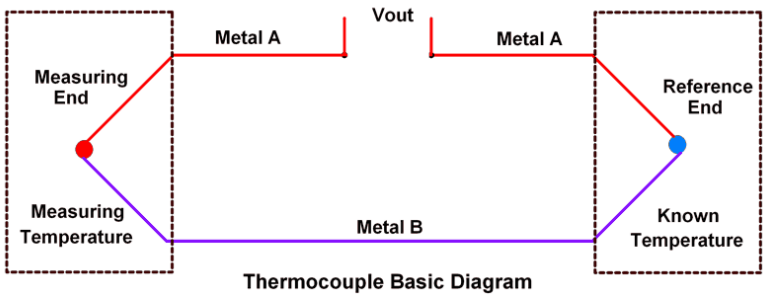
- Interfacing and coding Thermocouple
- Interfacing and coding LCD Display
- Designing 3D Design
- Designing PCB

Thermocouple

A study of the project's temperature requirements revealed that a sensor that could measure a high temperature range was needed. So, type K thermocouple probe type temperature sensor was chosen. It's can measure temperature range -40 to +1,260 °C, Model number is TC10-0

Thermocouple work principle

It is made by joining two dissimilar metals at one end and other end is either joined or not joined. The joined end is referred to as hot junction and other end is called as cold junction. The junction which temperature increases is called hot junction as well as measuring end and other end is called as cold junction or reference End. The cold junction will also have certain temperature according to the **Seebeck Effect.** The heat flows from the hot side to cold side, the temperature difference between these two junctions has created a small voltage and this voltage is used to measure the temperature and the temperature depends on thermocouple produced voltage.



Thermocouple Basic Diagram

Thermocouple created voltage is extremely small and it measured in terms of millivolt (one millivolt is equal to one thousandth of a volt).

Temperate measurement using example,

- If we had a thermocouple in a heat treat furnace and wanted to know what temperature it was in that furnace, we could attach a voltmeter to the cold junction and measure the voltage.
- That the furnace is operating at 1000 deg. F. and it is 100 deg. F at the cool end of the T/C. that a T/C measures the difference between the hot and cold junctions, the formula would be:

1000 (hot junction) - 100 (cold junction) = 900 deg. F.

- There seems to be a problem since we said that the furnace was at 1000 deg. F. This brings us to cold junction compensation.
- cold junction compensation is usually done automatically by the measuring instrument. The instrument measures the temperature at the cold junction and adds it back to the equation.

1000 (hot junction) - 100 (cold junction) = 900 deg. F + 100 deg. F (cold junction temp) = 1000 deg F

- This way the instrument indicates the actual temperature of the hot junction.

The thermocouple generated mini voltage rang voltage. These con not be read to microcontroller directly, AD595 is a complete instrumentation amplifier (Monolithic Thermocouple Amplifiers) with a Cold Junction Compensation. AD595 is compatible with a K-type thermocouple, it combines ice point reference with the pre-calibrated amplifier to produce a high-level output (10mV/ºC) directly from the thermocouple output. AD595 gain trimmed to match transfer characteristic of K-type thermocouple at 25ºC. The output of a K-type thermocouple in this temperature range is 40.44uV/ºC. The resulting gain for AD595 is 247.3 (10mV/ºC divided by 40.44uV/ºC). The input offset voltage for AD595 is 11uV, this offset arises because the AD595 is trimmed for a 250-mV output while applying a 25°C thermocouple input.

The output of AD595 is,

AD595 Output = (Type K Voltage + 11 uV) x 247.3

What for used thermocouple

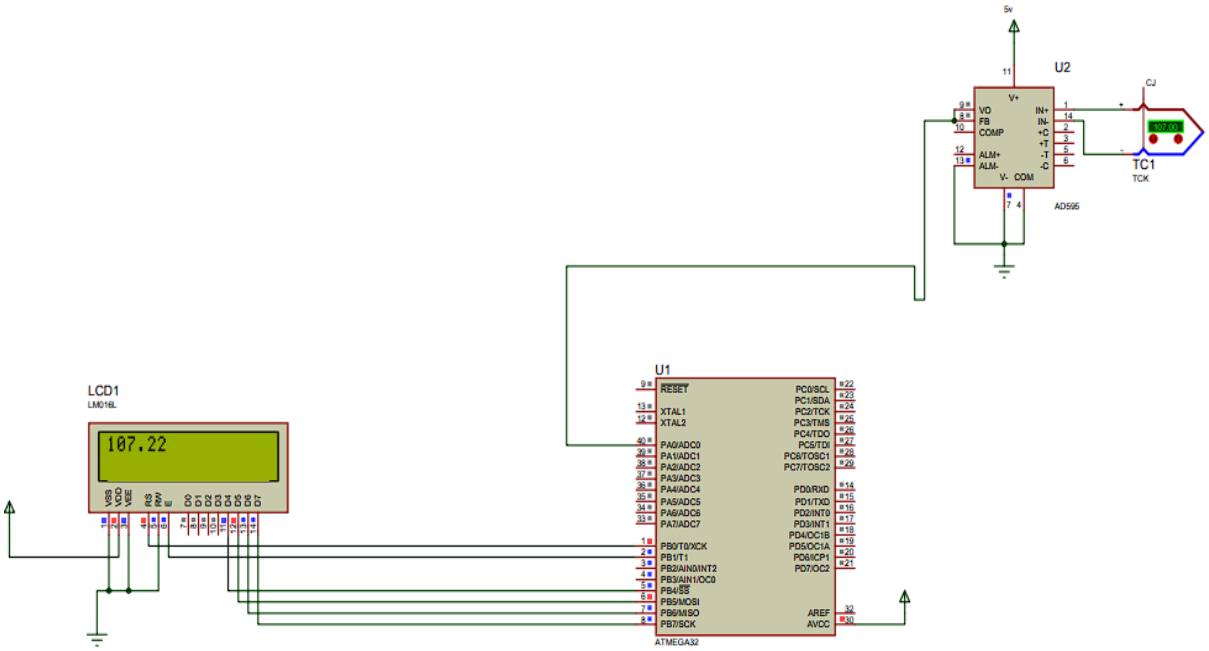
Kithul jaggery and Kithul treacle are used to make Kithul sap. Kithul sap produces jaggery and kithul Treacle when heated to a high temperature. To produce Kitul jaggery it is necessary to heat about 200 deg c and to produce Kitul honey it is necessary to heat about 103 deg c.

Accordingly, the Kitul sap should be heated above the boiling temperature. The solution in this process uses temperature sensors to read the temperature. The probe of the temperature sensor is in the container. It constantly collides with the liquid and reads the temperature and delivers it to the microchip

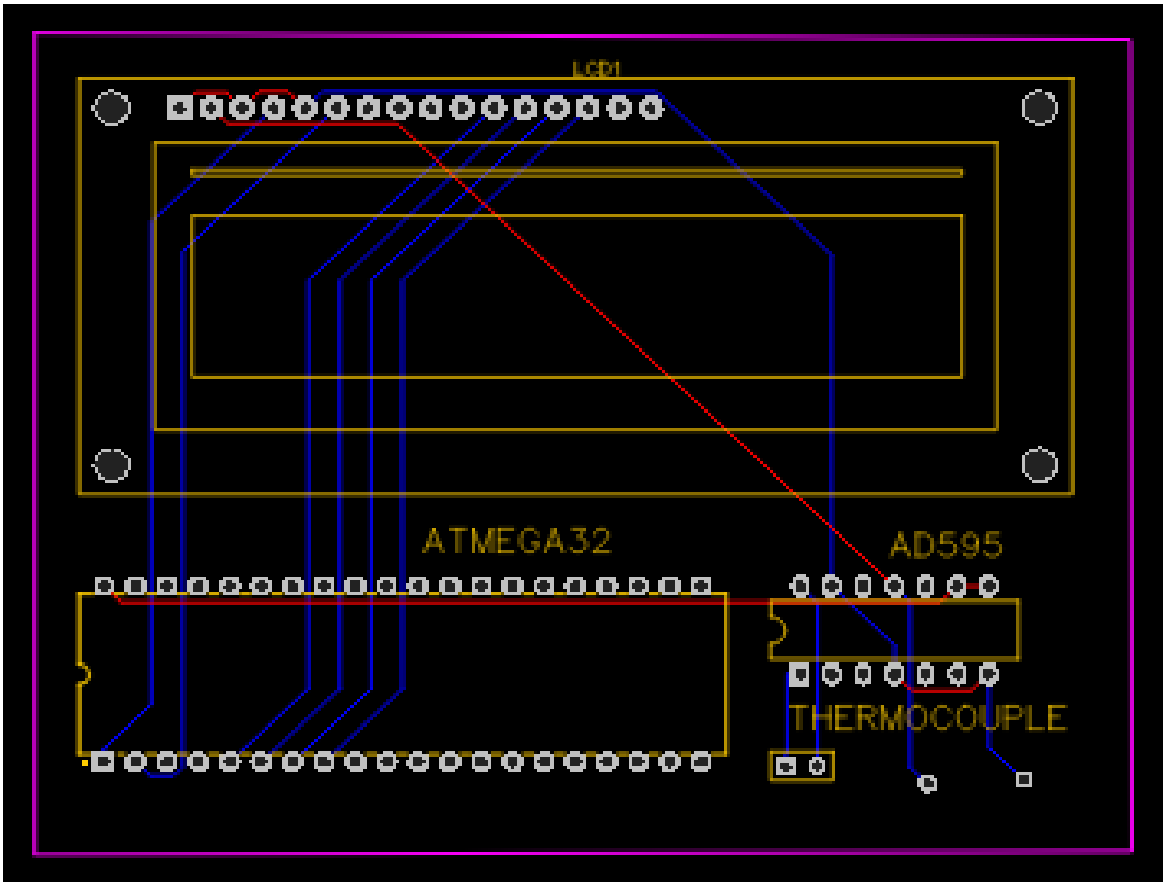
Thermocouple ia analog device so we need to convert that signal to digital. We use analog to digital conversion to these. Microcontroller port A server as analog to digital convert, so we connect port A to connect thermocouple to microcontroller

After the user select which product should be prepared, the temperature sensor starts to measure the temperature continuously

Circuit (Schematic Diagram)



PCB design



Code

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>



////////////////LCD display///////////////

#define LCD_Dir DDRB                                /* Define LCD data port direction */
#define LCD_Port PORTB                                      /* Define LCD data port */
#define RS PB0                                                      /* Define Register
Select (data reg./command reg.) signal pin */
#define EN PB1                                                      /* Define Enable signal
pin */


void LCD_Command( unsigned char cmnd )
{
       LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
       LCD_Port &= ~ (1<<RS);                                 /* RS=0, command reg. */
       LCD_Port |= (1<<EN);                        /* Enable pulse */
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);

       _delay_us(200);

       LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
       LCD_Port |= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);
       _delay_ms(2);
}


void LCD_Char( unsigned char data )
{
       LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
       LCD_Port |= (1<<RS);                        /* RS=1, data reg. */
       LCD_Port|= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);

       _delay_us(200);

       LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
       LCD_Port |= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);
       _delay_ms(2);
}

void LCD_Init (void)                                /* LCD Initialize function */
{
       LCD_Dir = 0xFF;                                             /* Make LCD command port
direction as o/p */
```

```
	_delay_ms(20);                                          /* LCD Power ON delay
always >15ms */

	LCD_Command(0x33);
	LCD_Command(0x32);                      /* send for 4 bit initialization of
LCD  */
	LCD_Command(0x28);                /* Use 2 line and initialize 5*7 matrix in
(4-bit mode)*/
	LCD_Command(0x0c);                /* Display on cursor off*/
	LCD_Command(0x06);                /* Increment cursor (shift cursor to right)*/
	LCD_Command(0x01);                /* Clear display screen*/
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
}


void LCD_String (char *str)                     /* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)                          /* Send each char of string
till the NULL */
	{
		LCD_Char (str[i]);
	}
}

void LCD_String_xy (char row, char pos, char *str)      /* Send string to LCD with xy
position */
{
	if (row == 0 && pos<16)
	LCD_Command((pos & 0x0F)|0x80);           /* Command of first row and required
position<16 */
	else if (row == 1 && pos<16)
	LCD_Command((pos & 0x0F)|0xC0);           /* Command of first row and required
position<16 */
	LCD_String(str);                                /* Call LCD string function */
}

void LCD_Clear()
{
	LCD_Command (0x01);                             /* Clear display */
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
}
//Thremocuple

char array[10];
int ADC_value;
float Temperature;

// configure the thermocouple sensor
void ADC_Init()
{
	//DDRA &= ~(1<<3);    /* Make ADC port as input */
	DDRA &= 0b10000000;
	ADCSRA = 0x87; /* Enable ADC, fr/128  */
}

int ADC_Read()
{
```

```
	ADMUX = 0x40;          /* Vref: Avcc, ADC channel: 0 */
	ADCSRA |= (1 << ADSC); /* Start conversion */
	while ((ADCSRA & (1 << ADIF)) == 0);  /*monitor end of conversion interrupt
flag */
	ADCSRA |= (1 << ADIF); /* Set the ADIF bit of ADCSRA register */
	return (ADCW);         /* Return the ADCW */
}




void measure_tempurature()
{

	ADC_Init(); /* Initialize the ADC */

	ADC_value = ADC_Read(); /* store the analog data on a variable */

	/* convert analog voltage into ºC and subtract the offset voltage */

	Temperature = (((ADC_value * 4.88)-0.0027) / 10.0)-26;
	dtostrf(Temperature, 3, 2, array);
}


int main()
{

	LCD_Init();
	LCD_Command(0xc1);
	LCD_String("welcome");
	_delay_ms(1000);
	LCD_Clear();

	while(1)
	{
		measure_tempurature();
		LCD_Clear();
		LCD_Command(0x80);
		LCD_String(array);
		_delay_ms(200);

	}
}
```

1. LCD Display

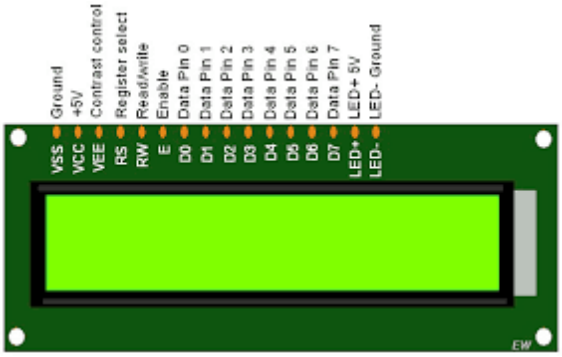


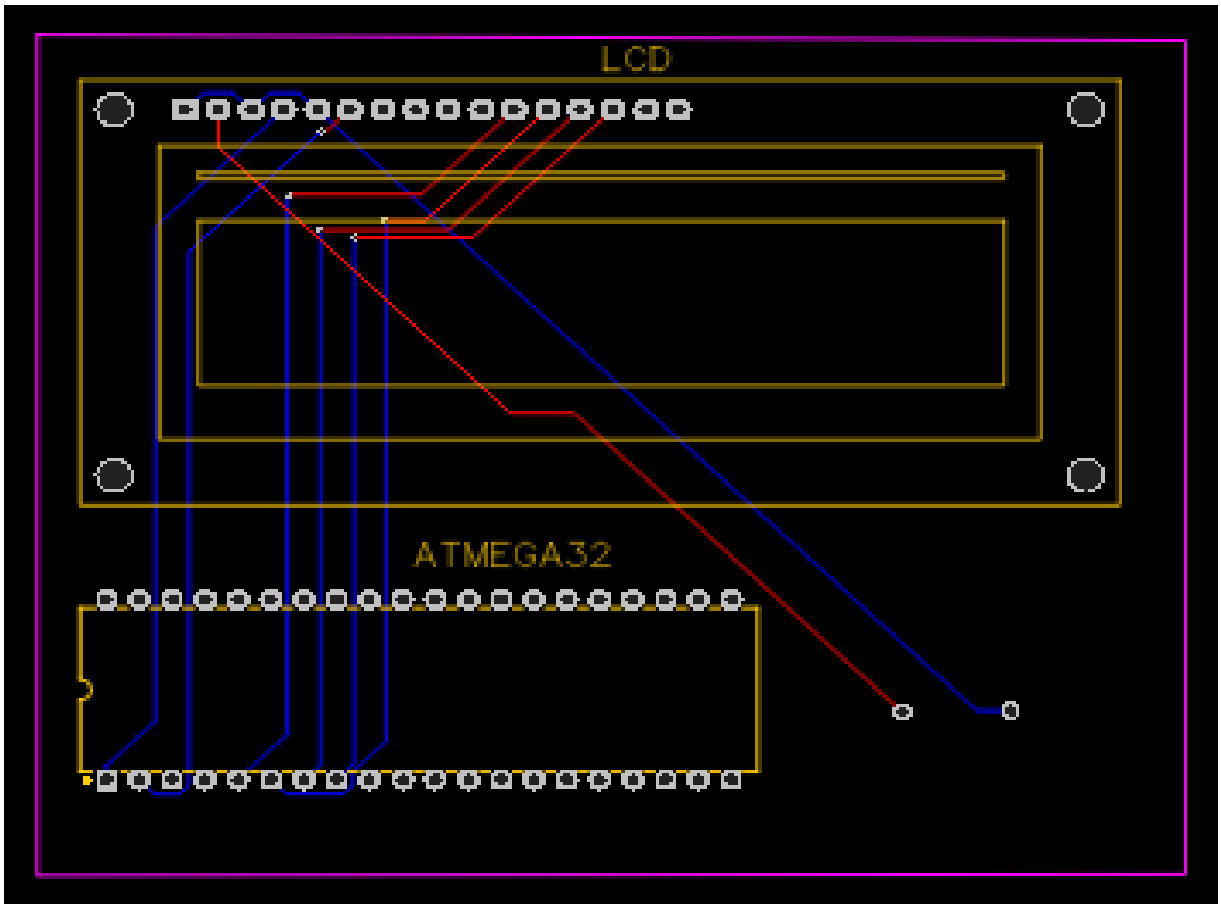
Figure 12 LCD Display

LCD means liquid display and it commonly used in circuit. These devices are thinner as we power consumption is extremely less. The operating voltage of this LCD is 4.7V-5.3V. It includes two rows each row can produce 16 Characters and very character can be built with 5*8 pixel matric in five dots are horizontally and dots are vertically. We can show a-z, A-Z,0-9, and special characters using LCD Display. The LCD works on the principle of **blocking light**. While constructing the LCDs, a reflected mirror is arranged at the back. An electrode plane is made of indium-tin-oxide which is kept on top and a polarized glass with a polarizing film is also added on the bottom of the device.

Display is used to show messages to the user about the processes of the machine

Circuit (Schematic Diagram)

PCB design



Code

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>



////////////////LCD display///////////////

#define LCD_Dir DDRB                                /* Define LCD data port direction */
#define LCD_Port PORTB                                      /* Define LCD data port */
#define RS PB0                                                      /* Define Register
Select (data reg./command reg.) signal pin */
#define EN PB1                                                      /* Define Enable signal
pin */


void LCD_Command( unsigned char cmnd )
{
       LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
       LCD_Port &= ~ (1<<RS);                                /* RS=0, command reg. */
       LCD_Port |= (1<<EN);                        /* Enable pulse */
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);

       _delay_us(200);

       LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
       LCD_Port |= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);
       _delay_ms(2);
}


void LCD_Char( unsigned char data )
{
       LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
       LCD_Port |= (1<<RS);                        /* RS=1, data reg. */
       LCD_Port|= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);

       _delay_us(200);

       LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
       LCD_Port |= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);
       _delay_ms(2);
}

void LCD_Init (void)                                /* LCD Initialize function */
{
       LCD_Dir = 0xFF;                                             /* Make LCD command port
direction as o/p */
```

```
	_delay_ms(20);                                          /* LCD Power ON delay
always >15ms */

	LCD_Command(0x33);
	LCD_Command(0x32);                      /* send for 4 bit initialization of
LCD  */
	LCD_Command(0x28);                /* Use 2 line and initialize 5*7 matrix in
(4-bit mode)*/
	LCD_Command(0x0c);                /* Display on cursor off*/
	LCD_Command(0x06);                /* Increment cursor (shift cursor to right)*/
	LCD_Command(0x01);                /* Clear display screen*/
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
}


void LCD_String (char *str)                     /* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)                          /* Send each char of string
till the NULL */
	{
		LCD_Char (str[i]);
	}
}

void LCD_String_xy (char row, char pos, char *str)      /* Send string to LCD with xy
position */
{
	if (row == 0 && pos<16)
	LCD_Command((pos & 0x0F)|0x80);           /* Command of first row and required
position<16 */
	else if (row == 1 && pos<16)
	LCD_Command((pos & 0x0F)|0xC0);           /* Command of first row and required
position<16 */
	LCD_String(str);                                /* Call LCD string function */
}

void LCD_Clear()
{
	LCD_Command (0x01);                             /* Clear display */
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
}
//Thremocuple

char array[10];
int ADC_value;
float Temperature;

// configure the thermocouple sensor
void ADC_Init()
{
	//DDRA &= ~(1<<3);    /* Make ADC port as input */
	DDRA &= 0b10000000;
	ADCSRA = 0x87; /* Enable ADC, fr/128  */
}

int ADC_Read()
{
```

```
	ADMUX = 0x40;          /* Vref: Avcc, ADC channel: 0 */
	ADCSRA |= (1 << ADSC); /* Start conversion */
	while ((ADCSRA & (1 << ADIF)) == 0);  /*monitor end of conversion interrupt
flag */
	ADCSRA |= (1 << ADIF); /* Set the ADIF bit of ADCSRA register */
	return (ADCW);         /* Return the ADCW */
}




void measure_tempurature()
{

	ADC_Init(); /* Initialize the ADC */

	ADC_value = ADC_Read(); /* store the analog data on a variable */

	/* convert analog voltage into ºC and subtract the offset voltage */

	Temperature = (((ADC_value * 4.88)-0.0027) / 10.0)-26;
	dtostrf(Temperature, 3, 2, array);
}


int main()
{

	LCD_Init();
	LCD_Command(0xc1);
	LCD_String("welcome");
	_delay_ms(1000);
	LCD_Clear();

	while(1)
	{
		measure_tempurature();
		LCD_Clear();
		LCD_Command(0x80);
		LCD_String(array);
		_delay_ms(200);

	}
}
```

10.2 205093D - SALAMA F.H. F

Responsible Part

- ❖ Interfacing and coding weight sensor
- ❖ Interfacing Power supply
- ❖ Complete the 3D Animation
- ❖ Designing PCB

1. Load Cell

A straight bar Load Cell is used in our project to measure the weight of the sap poured. This load cell is attached under the filtering container to measure the weight accurately.

- ➢ It will be measuring the weight continuously and once 4kg of sap is measured, the buzzer will ring to indicate the user to stop poring sap since it reached the limit.

According to functions and requirements of the project, I choose to use **TAL220B MINIATURE LOAD CELL** which is inexpensive yet reliable and can measure up to 5kg of weight.

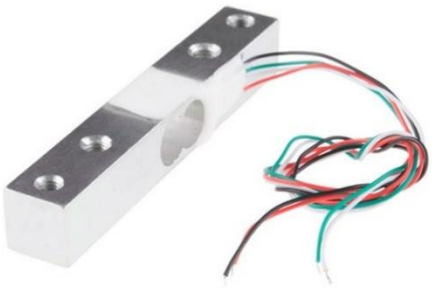


Figure 13 Loadcell

Technique and Specification

Load cells are designed to measure a specific force, and ignore other forces being applied. Load cell convert the load acting on them into electrical signals. The measuring is done with very small resistor patterns called strain gauges. The gauges are bonded onto a beam that deforms when weight is applied, in turn deforming the strain-gauge. As the strain gauge is deformed, it's electrical resistance changes in proportion to the load. The load cell we have chosen, use 4 strain gauges, some in compression, some under tension, which maximizes the sensitivity of the load cell, and automatically cancels the effect of temperature.

The electrical signal output by the load cell is very small and requires specialized amplification. And I have used the HX711 amplifier for amplification and measurement of the electrical output.

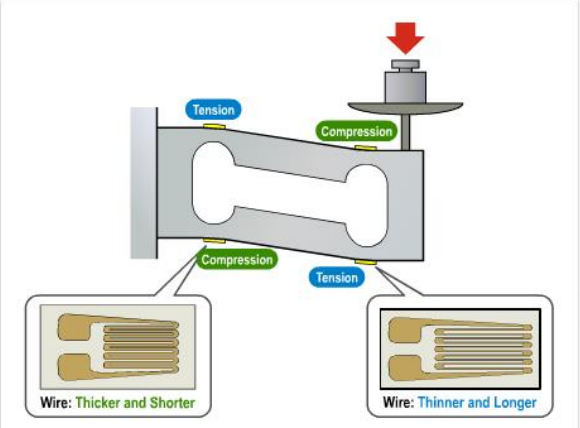
Features:

- Measures up to 5kg weight
- Material: aluminum-alloy
- Type: Parallel beam type (Strain gauge)
- Automatically cancels the effect of temperature
- Works under lower current requirements.

Component	TAL220B load cell
Type	Strain gauge (Parallel beam type)
Rated Output	1.0±0.15 mv/V
Excitation Voltage	3 to 10 Vdc
Operating temperature range	-20 to ~+55°C
Electrical connection	Cable 4 color wire, Ø 0.8 × 200 mm
	Excitation (+): Red Signal (+): Green Excitation (-): Black Signal (-): White

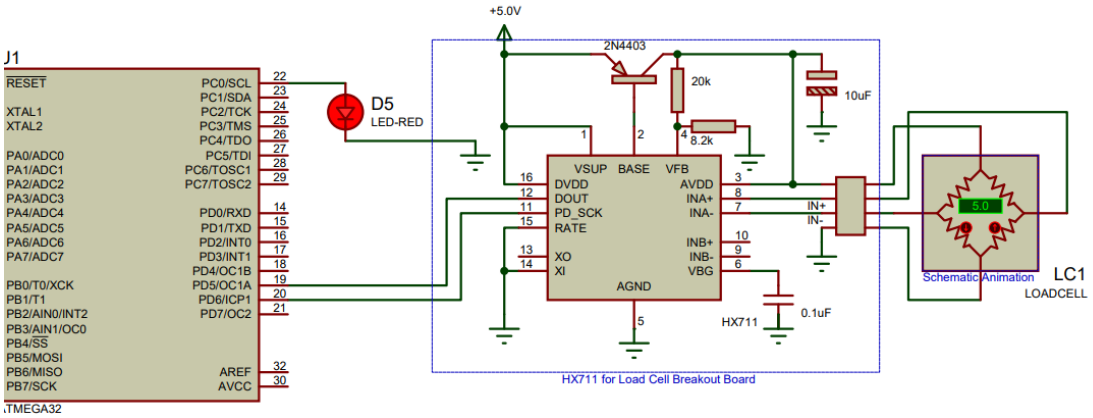
Working principle

When the load is applied to the body of a resistive load cell as shown in the figure, the elastic member, deflects as shown and creates a strain at those locations due to the stress applied. As a result, two of the strain gauges are in compression, whereas the other two are in tension. During a measurement, weight acts on the load cell's metal spring element and causes elastic deformation.

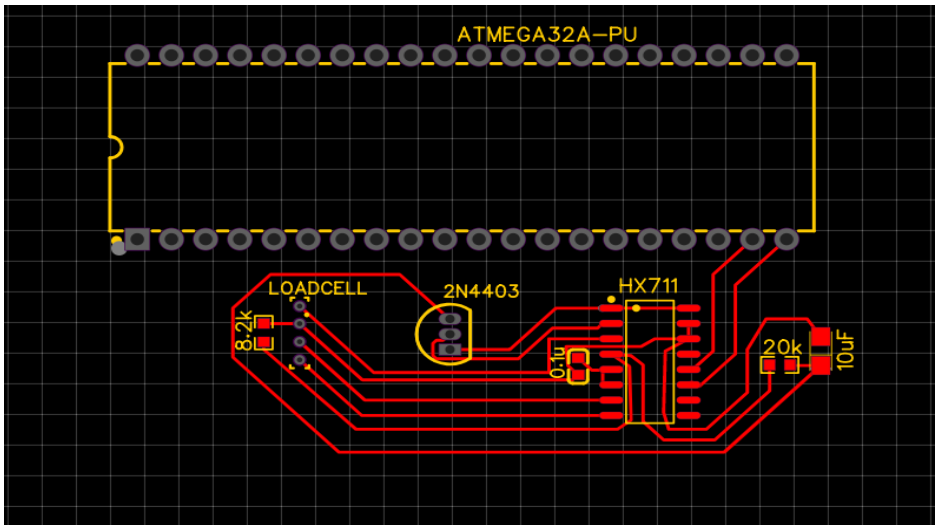


This strain (positive or negative) is converted into an electrical signal by a strain gauge installed on the spring element. We use wheatstone bridge circuit to convert this change in strain/resistance into voltage which is proportional to the load.

Circuit



PCB design



Code

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t hx711H=0; //Load Scale High Bits
uint16_t hx711L=0; //Load Scale Low Bits

////////////////LCD display///////////////

#define LCD_Dir DDRB                                /* Define LCD data port direction */
#define LCD_Port PORTB                                      /* Define LCD data port */
#define RS PB0                                                  /* Define Register
Select (data reg./command reg.) signal pin */
#define EN PB1                                                  /* Define Enable signal
pin */

void LCD_Command( unsigned char cmnd )
{
	LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
	LCD_Port &= ~ (1<<RS);                                  /* RS=0, command reg. */
	LCD_Port |= (1<<EN);                        /* Enable pulse */
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}


void LCD_Char( unsigned char data )
{
	LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
	LCD_Port |= (1<<RS);                        /* RS=1, data reg. */
	LCD_Port|= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}

void LCD_Init (void)                            /* LCD Initialize function */
{
	LCD_Dir = 0xFF;                                         /* Make LCD command port
direction as o/p */
```

```
	_delay_ms(20);                                          /* LCD Power ON delay
always >15ms */

	LCD_Command(0x33);
	LCD_Command(0x32);                      /* send for 4 bit initialization of
LCD  */
	LCD_Command(0x28);              /* Use 2 line and initialize 5*7 matrix in
(4-bit mode)*/
	LCD_Command(0x0c);              /* Display on cursor off*/
	LCD_Command(0x06);              /* Increment cursor (shift cursor to right)*/
	LCD_Command(0x01);              /* Clear display screen*/
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
}


void LCD_String (char *str)                     /* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)                          /* Send each char of string
till the NULL */
	{
		LCD_Char (str[i]);
	}
}

void LCD_String_xy (char row, char pos, char *str)      /* Send string to LCD with xy
position */
{
	if (row == 0 && pos<16)
	LCD_Command((pos & 0x0F)|0x80);         /* Command of first row and required
position<16 */
	else if (row == 1 && pos<16)
	LCD_Command((pos & 0x0F)|0xC0);         /* Command of first row and required
position<16 */
	LCD_String(str);                                /* Call LCD string function */
}

void LCD_Clear()
{
	LCD_Command (0x01);                             /* Clear display */
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
}


float loadCellRead();
void Led(bool x);

float hx=0;

int main(void)
{
	//DDRD|=(1<<6); //Load cell clock pin
	//PORTD&=~(1<<6);//Clock pin low
	DDRC |= (1<<PC0); // buzzer
	DDRD|= 0xff; //Load cell clock pin
	PORTD&=~(1<<6);//Clock pin low

	LCD_Init();
```

```
        LCD_Command(0xc1);
        LCD_String("welcome");
        _delay_ms(1000);
        LCD_Clear();

        while(1){

                hx=loadCellRead();
                char buf[100];
                dtostrf(hx, 2, 2, buf);
                LCD_Clear();

                if (hx >= 5)
                {
                        LCD_String("container filled");
                        Led(true);
                        _delay_ms(1000);
                        LCD_Clear();
                }
                else{
                        LCD_String(buf);
                        Led(false);
                        _delay_ms(1000);
                        LCD_Clear();
                }
        }
}

float loadCellRead(){
        hx711H=0;hx711L=0;  //clear variables
        for(uint8_t i=0;i<8;i++){  // Load cell data high 8 bits
                PORTD|=(1<<6); //Clock pin high
                _delay_us(10);
                if ((PIND&(1<<5))>>5)  //read data pin
                {hx711H|=(1<<(7-i));//set hx 711 varible
                }
                else
                {hx711H&=~(1<<(7-i));
                }
                PORTD&=~(1<<6); //Clock pin low
                _delay_us(5);
        }


        for(uint8_t i=0;i<16;i++){ // Load cell data low 16 bits
                PORTD|=(1<<6); //Clock pin high
                _delay_us(10);
                if ((PIND&(1<<5))>>5) //read data pin
                {hx711L|=(1<<(15-i));
                }
                else
                {hx711L&=~(1<<(15-i));
                }
                PORTD&=~(1<<6); //Clock pin low
                _delay_us(5);
        }

        hx711L=hx711L>>1; //shift bits

        if (hx711H&1)  //bit setup
        {hx711L|=(1<<15);
        }
```

```
        else
        {hx711L&=~(1<<15);
        }
        hx711H=hx711H>>1;

        return (hx711H*(65536/18029.6))+hx711L/18029.6; //load cell calibration
}

void Led(bool x)
{
        if (x == true)
        {
                PORTC |= (1 << PINC0);
        }
        else
        {
                PORTC &= ~(1 << PINC0);
        }
}
```

2. Power supply

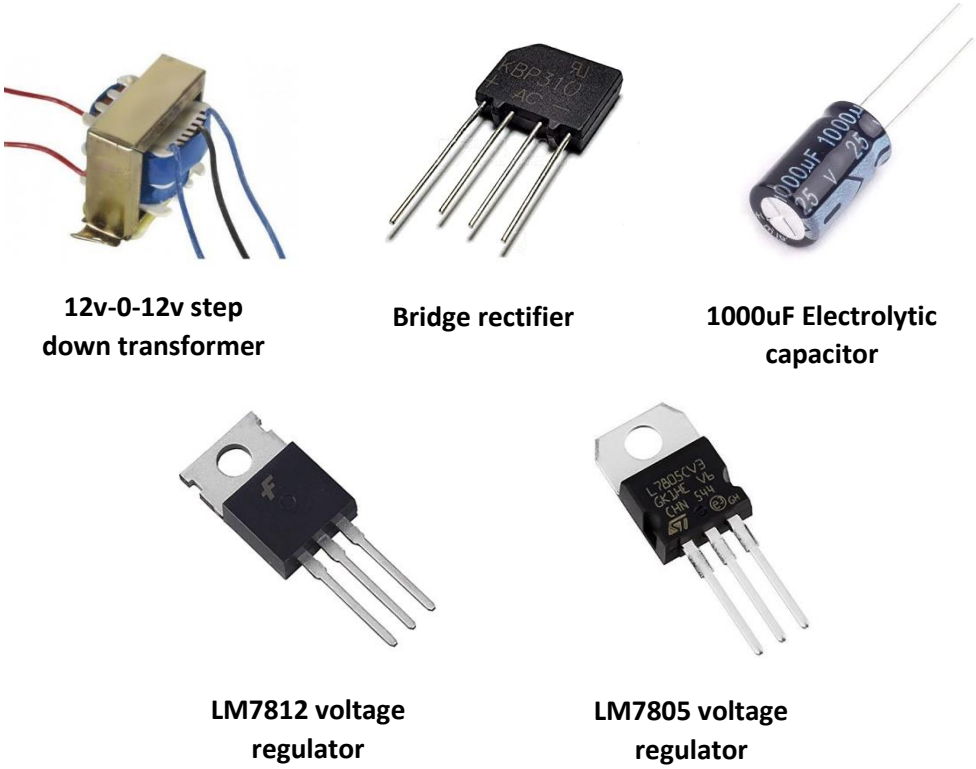


Figure 14 Power Supply

Technique and Specification

In DC (direct current), the voltage is always constant, and the electricity flows in a certain direction. In contrast, in AC (alternating current), the voltage periodically changes from positive to negative and from negative to positive, and the direction of the current also periodically changes accordingly. It should be noted that depending on cpmponent different type of supply should be provided. The supply which comes to households in Sri Lanka is 220V AC supply. But most of the components of our project requires DC supply. So, I have used the following method to convert AC to DC

Conversion from 220V AC to 12V and 5V DC

Rectifiers

Rectification is the process of conversion of AC supply to DC supply. Rectifiers are devices that convert the AC supply into the DC supply.
Basically, this conversion can be divided into four sub-steps:

1. Stepping down voltage

Generally, there is a high voltage AC supply as it is easy to transfer with minimum losses. However, our devices need a low voltage supply so use a step-down transformer for this

purpose. In step-down transformers, the primary coil has a higher number of loops as compared to the secondary coil.

2. AC to DC conversion

After Stepping down of voltage, AC is converted to DC using rectifiers. A full bridge rectifier is used to convert AC to DC. In this device, 4 diodes are used which operates in forwarding bias and not in reverse bias. During the positive half cycle two of the diodes are operated and during the negative half cycle other two diodes and operated. This way AC supply is rectified to the DC supply.

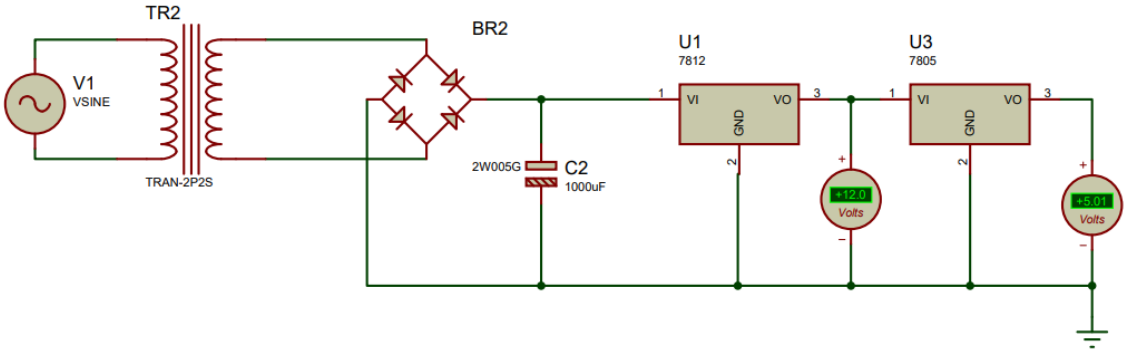
3. Purifying DC waveforms

The DC waveforms generated in the above step do not have pure DC waveforms. It is in form of pulses and has fluctuating supply. Capacitors are devices that are used to do this task. The capacitor is used to store energy when the input voltage is increasing from zero to the highest value. The energy from the capacitor can be discharged when the input voltage is decreasing to zero. This straightens the waveforms to a good extent.

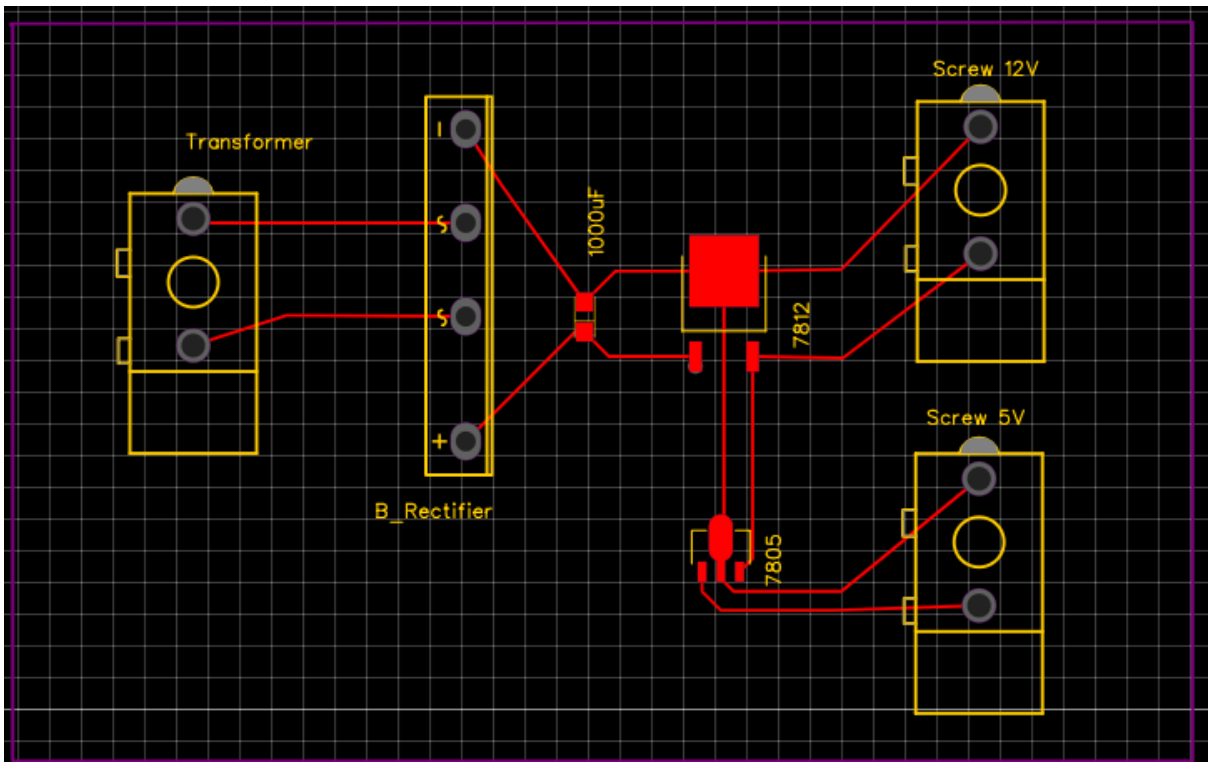
4. Fixing up DC voltage

Finally, DC voltage is converted to the fixed desired value by using the voltage regulator IC. The DC voltage regulators IC consists of an Integrated Circuit that finally converts our DC supply to a given voltage. For example, to convert to a 5V DC supply, we use the 7805 Voltage Regulator IC. And to convert to a 12V DC supply, we use a 7812 Voltage regulator IC.

Circuit



PCB design


Transformer
B_Rectifier
1000uF
7812
7805
Screw 12V
Screw 5V

10.3 205108G - THENNAKOON T.M.M. S

Responsible Part

- Interfacing and coding Stepper Motor
- Interfacing and coding Key Pad
- Designing 3D animation
- Designing PCB

1.Stepper Motor

In our project we are using stepper motor to spoon the sap until the kithul treacle or kithul jaggery is made. The handle of spoon is connected to the motor, so the stepper motor is placed above the sap container to mix sap properly.

In our project the stepper motor will activate under following functions.

- When the container is filled with sap, motor stars to rotate and it continues until the temperature set to set temperature
- When the user increases the set temperature after the process is done, motor starts again rotating until temperature set with the new set temperature.

To select a motor, we consider the below requirements:

- In our project we spoon 4kg sap for each time, so the motor requires to be able to hold that weight.
- We are expecting to produce kithul jaggery, and it is very thick and heavy. For making kithul treacle or jaggery, motor requires high torque.

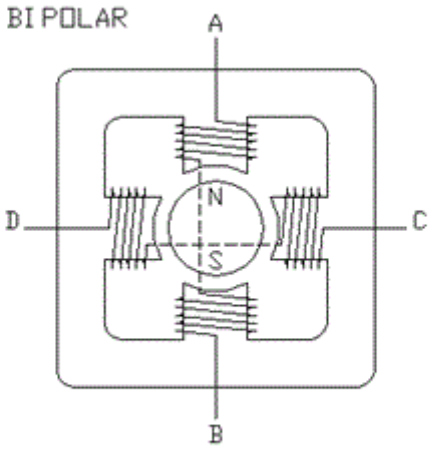
According to functions and requirements of the project, I choose to use **NEMA 23 (size 57mm) Hybrid Stepper Motor** with **L293D** Driver. This stepper motor is a high torque hybrid stepping motor.

Figure 15 Stepper Motor

Technique and Specification

NEMA 23 bipolar Stepper Motor

This bipolar stepper motor has one winding per stator phase. A two-phase bipolar stepper motor will have 4 leads. In a bipolar stepper, we don't have a common lead like in a unipolar stepper motor.



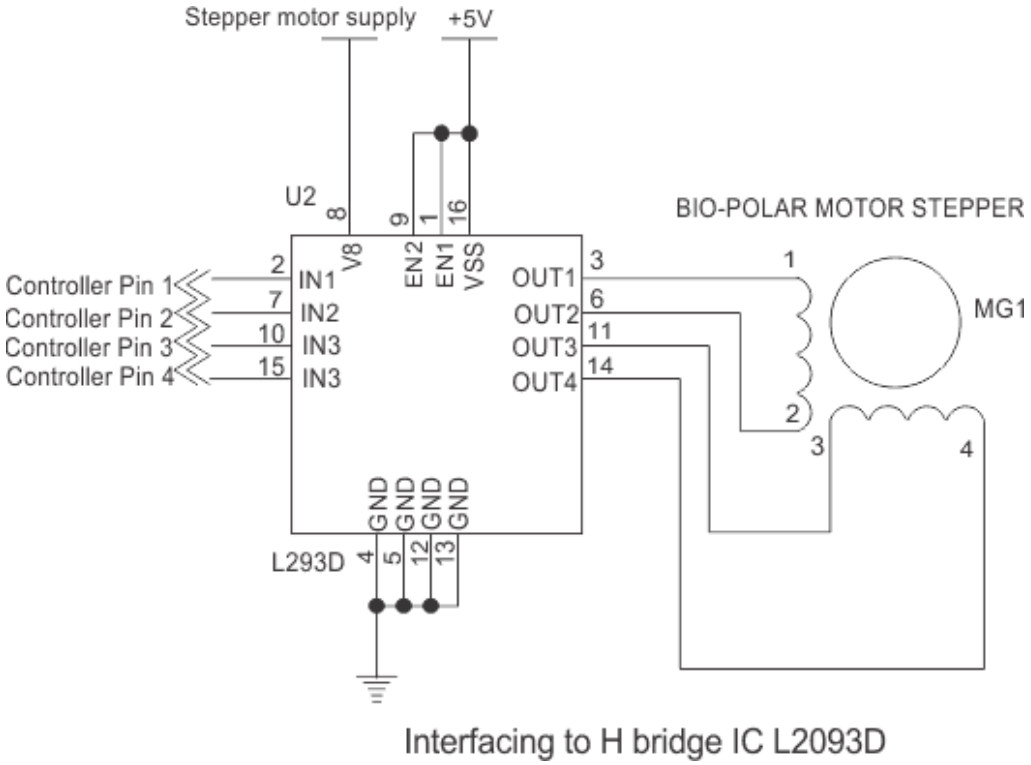
Bipolar Stepper winding arrangement showing stator and rotor

To drive a bipolar stepper, we need a driver IC with an internal H bridge circuit. This is because, to reverse the polarity of stator poles, the current needs to be reversed. This can only be done through a H bridge.

There are two other reasons to use an H Bridge IC

1. The current drawn by a stepper motor is quite high. The micro-controller pin can only provide up to 15 mA at maximum. The stepper needs current which is around ten times this value. An external driver IC can handle such high currents.
2. H Bridge is used is that the stator coils are nothing but an inductor. When coil current changes direction a spike is generated. A normal micro-controller pin cannot tolerate such high spikes without damaging itself. Hence to protect micro-controller pins, an H bridge is necessary.

4 micro-controller pins are required to control the motor. We need to provide the L293D with a 5 V supply as well as the voltage at which the motor needs to operate. Since we will be using both the drivers of the IC, we will assert the enable pin for both.



Bipolar Motor Driver Circuit Interfacing Diagram

bipolar stepper motors are a little complex to wire as we have to use a current reversing H bridge driver IC like an L293D.

But the advantage is that the current will flow through the full coil. The resulting torque generated by the motor is larger as compared to a unipolar motor.

L293D

the drive must be able to supply sufficient current for your stepper. The micro-controller must only provide the step and direction signal to the drive.

The L293D devices is a quadruple high1 current half-H driver. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. This device is solenoid, DC, and bipolar stepping motor, as well as supply application.

Features:

• Wide Supply-Voltage Range: 4.5 V to 36 V

• Separate Input-Logic Supply

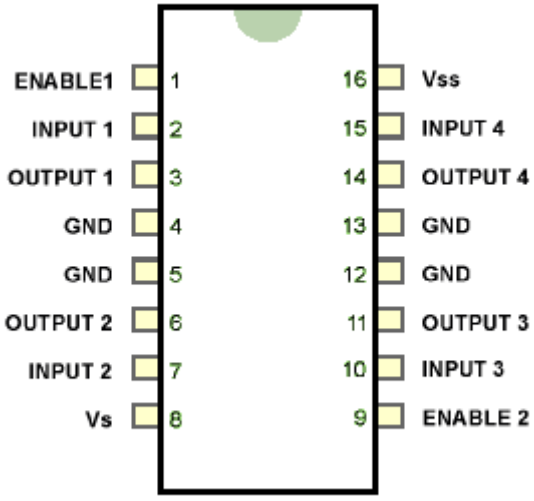
• Internal ESD Protection

• High-Noise-Immunity Inputs

• Output Current 1 A Per Channel (600 mA for designed to drive inductive loads such as relays, L293D)

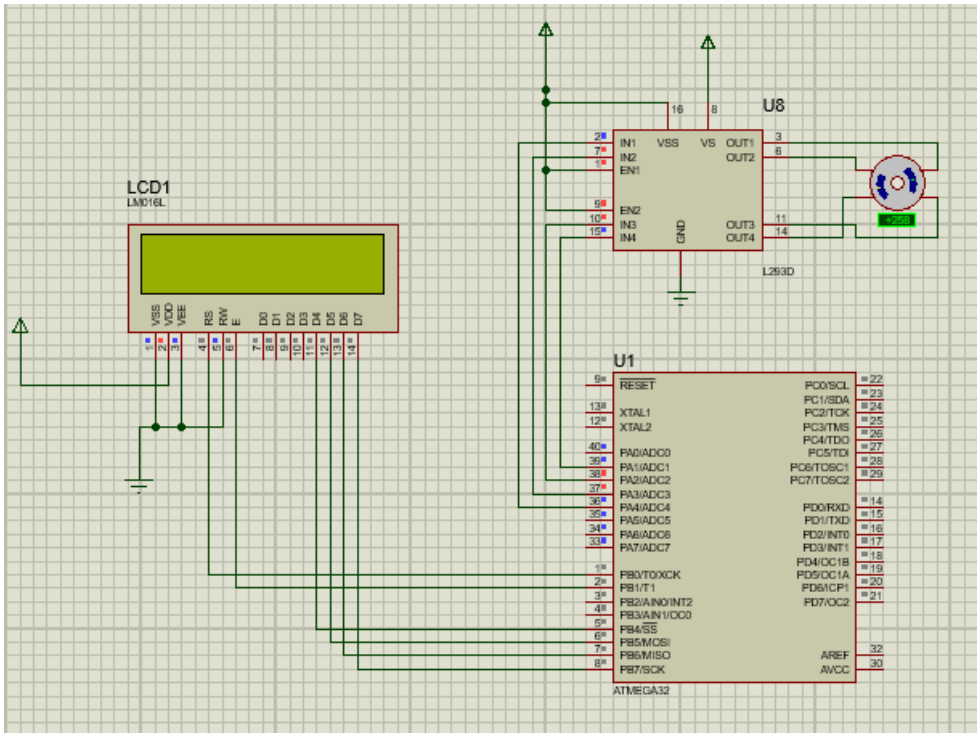
• Peak Output Current 2 A Per Channel (1.2 A for other high-current/high-voltage loads in positiveL293D)

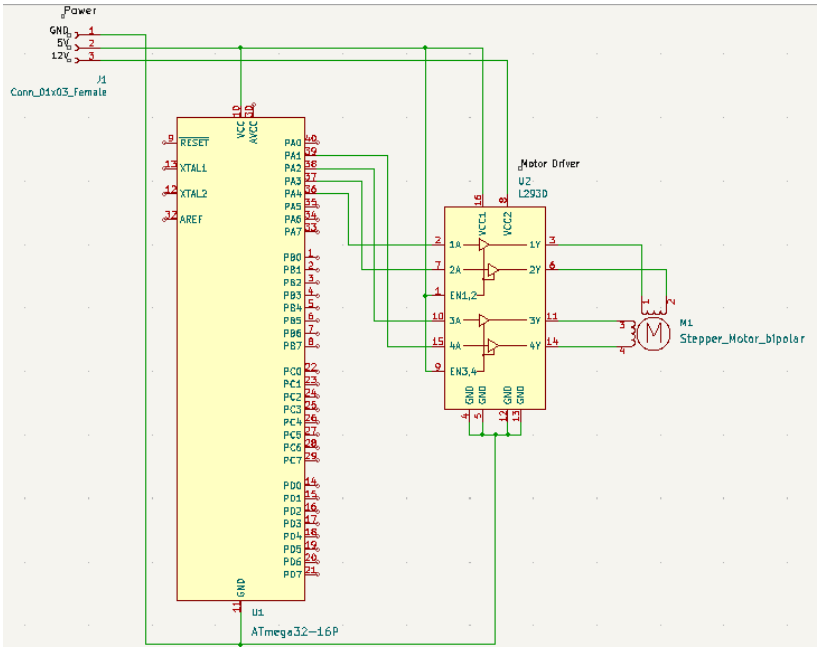
• Output Clamp Diodes for Inductive Transient Each output is a complete totem-pole drive circuit, Suppression (L293D)



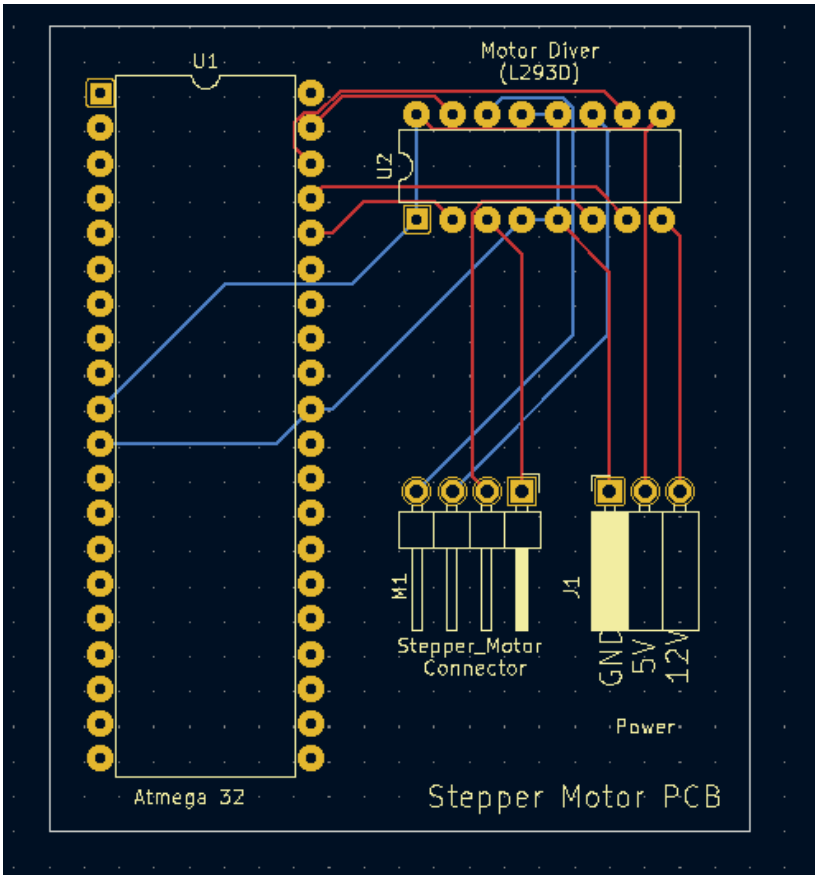
Pin diagram

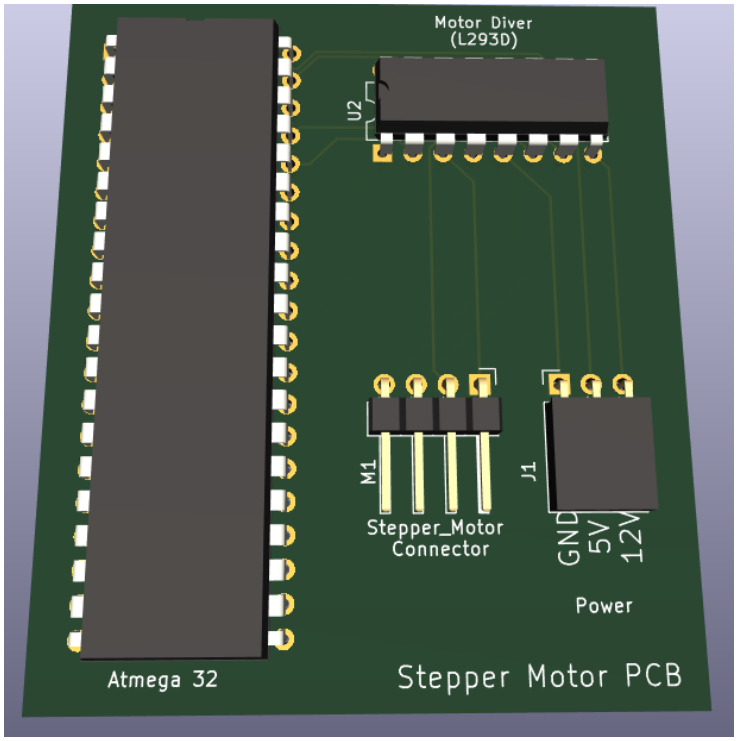
Circuit





PCB design


U1
Motor Diver
(L293D)
U2
M1
Stepper_Motor
Connector
J1
GND
5V
12V
Power
Atmega 32
Stepper Motor PCB


Motor Diver
(L293D)
U2
M1
Stepper_Motor
Connector
J1
GND
5V
12V
Power
Atmega 32
Stepper Motor PCB

Code

```
#define F_CPU 8000000UL

#include <avr/io.h>
#include <util/delay.h>

void stepper();

int main(void)
{
	DDRA = 0xFF;
	PORTA = 0x00;

    /* Replace with your application code */
    stepper();
}

void stepper(){
		//All pins of PORTC as output		//Initially all pins as
output high
	while(1)
	{
		PORTA = 0b00010010;
		_delay_ms(100);
		PORTA = 0b00000110;
		_delay_ms(100);
		PORTA = 0b00001100;
		_delay_ms(100);
		PORTA = 0b00011000;
		_delay_ms(100);
	}
}
```

2. Keypad

In our project we are using keypad to give options, customize the temperature and open/close valves. we place keypad in front of the container so that it is user friendly.

In our project the keypad will use under following functions.

- Selecting options by keypad makes the user go through the specific process of making kithul treacle or jaggery. When the user selects kithul treacle option or jaggery making option, system sets the defaults (temperature limit, valve no) for the specific product.
- And the user has the feature to increase the default set temperature to continue the process if the user is not satisfied with the output product. User can use 0 to 9 buttons to enter the new temperature.
- At the end, user has the options to open valve or continue the process by using keypad. And after getting the product out, user can close valves by using keypad.

According to functions and requirements of the project, I choose to use **4x4 Matrix Membrane Keypad (#27899)**. This 16-button keypad provides a useful human interface component for microcontroller projects. Convenient adhesive backing provides a simple way to mount the keypad in a variety of applications.

Figure 16 Keypad

Technique and Specification

4*4 keypad consists of 8 pins that can be identified as 4 input pins and 4 output pins. The interface of the keypad has 16 buttons which are placed between 4 rows and 4 columns. A keypress establishes a connection between the corresponding row and column where the switch is placed. To read the keypress, we need to configure the rows as outputs and columns as inputs. Columns are read after applying signals to the rows in order to determine whether or not a key pressed and if pressed, which key is pressed.

For example, say your program pulls all four columns low and then pulls the first row high. It then reads the input states of each column and reads pin 1 high. This means that a contact has been made between column 4 and row 1, so button 'A' has been pressed.

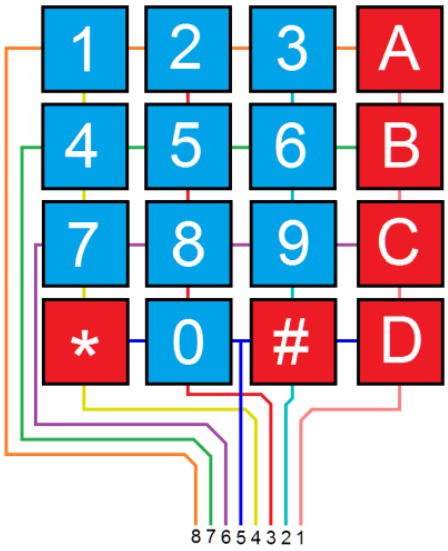
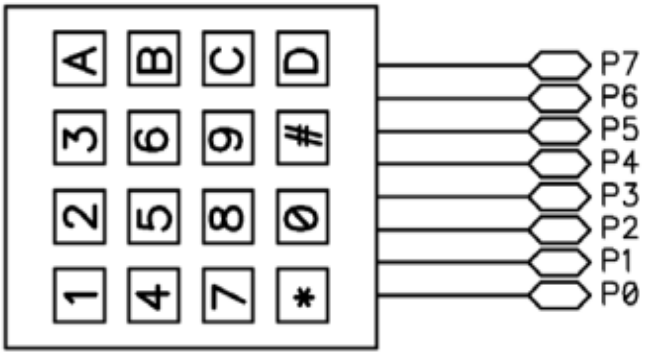


Figure 1: Matrix Keypad Connections



connection diagram

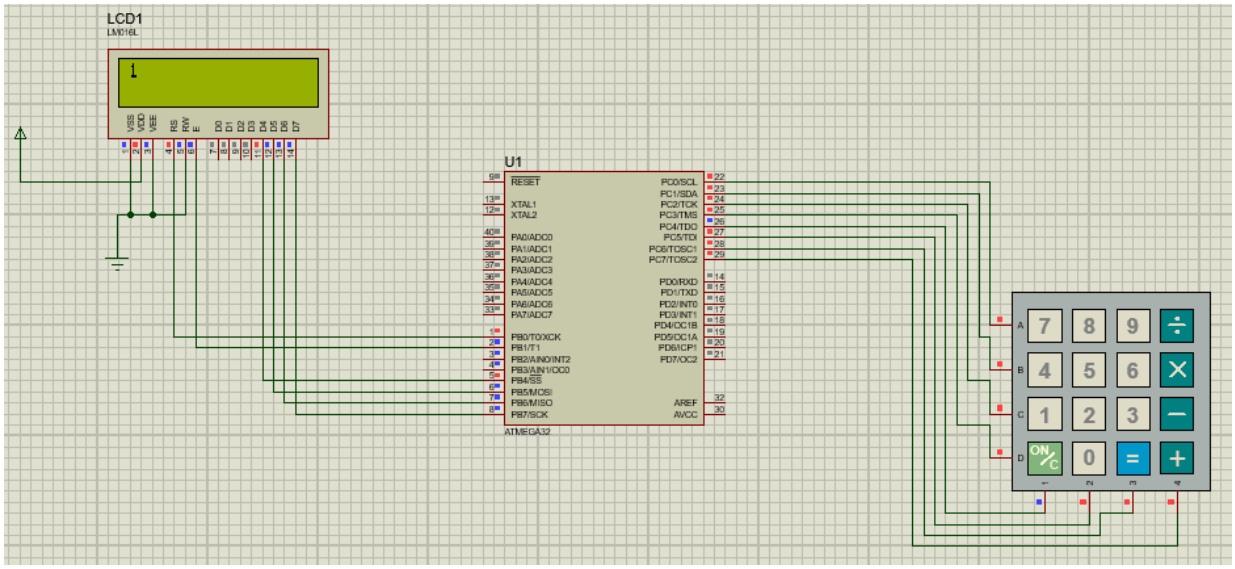
Key Features:

- Ultra-thin design & adhesive backing provides easy integration to any project
- Easy communication with any microcontroller
- Cable included

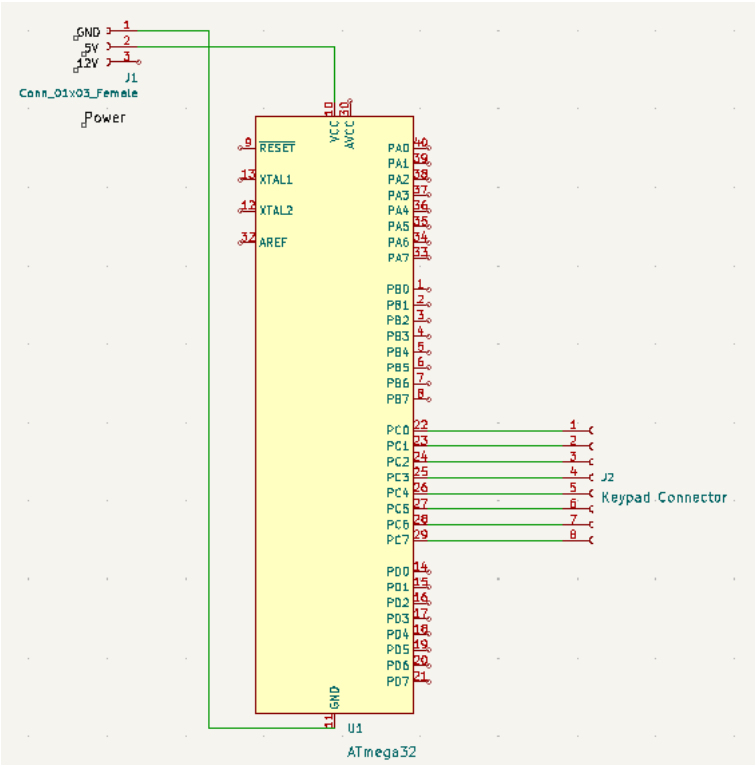
Details:

- Maximum Rating: 24 VDC, 30 mA
- Interface: 8-pin access to 4×4 matrix
- Dimensions: Keypad: 2.7 x 3.0 in (6.9 x 7.6 cm)
  Cable: 0.78 x 3.5 in (2.0 x 8.8 cm)
- Operating temp range: 32 to 122 °F (0 to 50 °C)

Circuit

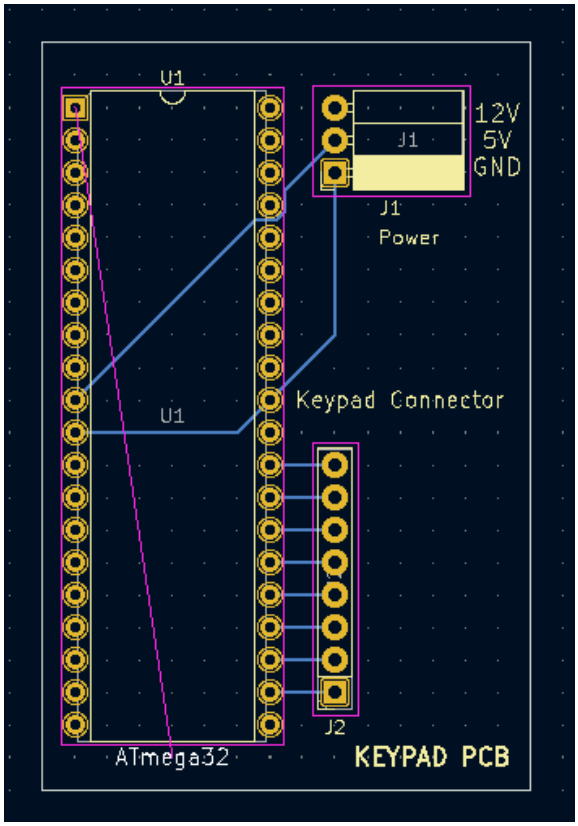


Keypad circuit design in protues



Keypad circuit design in kicad

PCB design


U1
12V
5V
GND
J1
J1
Power
U1
Keypad Connector
J2
ATmega32
KEYPAD PCB

Code

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include <avr/interrupt.h>


////////////////LCD display///////////////

#define LCD_Dir DDRB                                /* Define LCD data port direction */
#define LCD_Port PORTB                                      /* Define LCD data port */
#define RS PB0                                                  /* Define Register
Select (data reg./command reg.) signal pin */
#define EN PB1                                                  /* Define Enable signal
pin */

////////////////////////define key pad//////////////////
#define KEY_PRT PORTC
#define KEY_DDR DDRC
#define KEY_PIN PINC

unsigned char keypad[4][4] = {
       {'7', '4', '1', 'C'},
       {'8', '5', '2', '0'},
       {'9', '6', '3', '='},
       {'/', 'x', '-', '+'}};

       unsigned char colloc, rowloc;

       char keyfind();

void LCD_Command( unsigned char cmnd )
{
       LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
       LCD_Port &= ~ (1<<RS);                              /* RS=0, command reg. */
       LCD_Port |= (1<<EN);                        /* Enable pulse */
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);

       _delay_us(200);

       LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
       LCD_Port |= (1<<EN);
       _delay_us(1);
       LCD_Port &= ~ (1<<EN);
       _delay_ms(2);
}


void LCD_Char( unsigned char data )
{
       LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
```

```
	LCD_Port |= (1<<RS);                    /* RS=1, data reg. */
	LCD_Port|= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}

void LCD_Init (void)                        /* LCD Initialize function */
{
	LCD_Dir = 0xFF;                                 /* Make LCD command port
direction as o/p */
	_delay_ms(20);                                  /* LCD Power ON delay
always >15ms */

	LCD_Command(0x33);
	LCD_Command(0x32);                      /* send for 4 bit initialization of
LCD  */
	LCD_Command(0x28);              /* Use 2 line and initialize 5*7 matrix in
(4-bit mode)*/
	LCD_Command(0x0c);              /* Display on cursor off*/
	LCD_Command(0x06);              /* Increment cursor (shift cursor to right)*/
	LCD_Command(0x01);              /* Clear display screen*/
	_delay_ms(2);
	LCD_Command (0x80);                         /* Cursor 1st row 0th position
*/
}

void LCD_String (char *str)                 /* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)                  /* Send each char of string
till the NULL */
	{
		LCD_Char (str[i]);
	}
}

void LCD_String_xy (char row, char pos, char *str)     /* Send string to LCD with xy
position */
{
	if (row == 0 && pos<16)
	LCD_Command((pos & 0x0F)|0x80);         /* Command of first row and required
position<16 */
	else if (row == 1 && pos<16)
	LCD_Command((pos & 0x0F)|0xC0);         /* Command of first row and required
position<16 */
	LCD_String(str);                                /* Call LCD string function */
}

void LCD_Clear()
{
	LCD_Command (0x01);                             /* Clear display */
	_delay_ms(2);
	LCD_Command (0x80);                             /* Cursor 1st row 0th position
*/
```

```
}

int main(void)
{
	LCD_Init();
	LCD_Command(0xc1);
	LCD_String("welcome");
	_delay_ms(1000);
	LCD_Clear();

	 while (1)
	 {
		char a = keyfind();
		LCD_Char(a);
		_delay_ms(1000);
		LCD_Clear();
		LCD_Clear();
	 }
}

char keyfind()
{
	while (1)
	{
		KEY_DDR = 0xF0; /* set port direction as input-output */
		KEY_PRT = 0xFF;
		do
		{
			KEY_PRT &= 0x0F; /* mask PORT for column read only */
			asm("NOP");
			colloc = (KEY_PIN & 0x0F); /* read status of column */
		} while (colloc != 0x0F);
		do
		{
			do
			{
				_delay_ms(20);             /* 20ms key debounce time */
				colloc = (KEY_PIN & 0x0F); /* read status of column */
				} while (colloc == 0x0F);      /* check for any key press
*/
				_delay_ms(40);                 /* 20 ms key debounce time
*/
				colloc = (KEY_PIN & 0x0F);
			} while (colloc == 0x0F);
			/* now check for rows */
			KEY_PRT = 0xEF; /* check for pressed key in 1st row */
			asm("NOP");
			colloc = (KEY_PIN & 0x0F);
			if (colloc != 0x0F)
			{
				rowloc = 0;
				break;
			}
			KEY_PRT = 0xDF; /* check for pressed key in 2nd row */
			asm("NOP");
			colloc = (KEY_PIN & 0x0F);
			if (colloc != 0x0F)
			{
				rowloc = 1;
				break;
			}
			KEY_PRT = 0xBF; /* check for pressed key in 3rd row */
```

```
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if (colloc != 0x0F)
                {
                        rowloc = 2;
                        break;
                }
                KEY_PRT = 0x7F; /* check for pressed key in 4th row */
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if (colloc != 0x0F)
                {
                        rowloc = 3;
                        break;
                }
        }
        if (colloc == 0x0E)
        return (keypad[rowloc][0]);
        else if (colloc == 0x0D)
        return (keypad[rowloc][1]);
        else if (colloc == 0x0B)
        return (keypad[rowloc][2]);
        else
        return (keypad[rowloc][3]);
}
```

10.4 205096N - SAMARAKKODY S.A.T. S

Responsible Parts

- Interfacing and coding solenoid valve
- Interfacing and coding buzzer
- Contribution to reports and presentations
- Designing PCB

1.Buzzer

In our project we are using buzzer to give a signal/ message to user for specific purposes. The sound of the buzzer makes user awareness about specific functions. In our project the buzzer will activate under following functions;

- When the weight of the sap of Kithul Flower equals to 4kg so that user can't pour the sap anymore to filter
- When the mixture turns to set temperature so that user can know the heating is stopped

And we place the buzzer in filtering part so that it can't be affected from temperature.

According to functions and requirements of the project, I choose to use **Piezoelectric Buzzer (Pin terminal type PS12 PS1240P02BT)** which is inexpensive yet reliable and can produce a variety of pleasant sounds, and analog sound, intermittent sound under low power consumption.

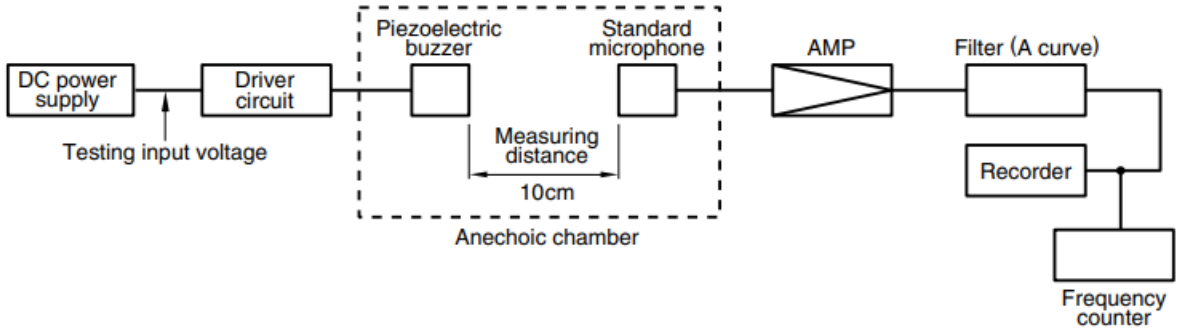
Figure 17 Buzzer

Technique and Specification

The Piezoelectric Buzzer is an electric device with a piezoelectric component used to produce sound. Piezoelectric components are made of a specific material type that help to convert electric energy into mechanical energy and mechanical energy into electric energy.

In short, the working principle of Piezoelectric Buzzer,

It works by applying an external voltage to the piezoelectric ceramic material. That voltage takes as an input signal that causes the piezoceramic to vibrate continuously and generates a pressure wave that the human ear can picks up as sound

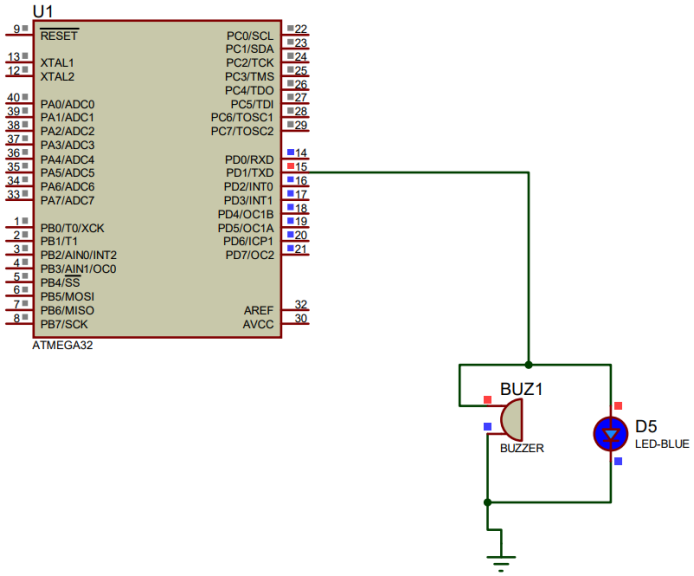


Features

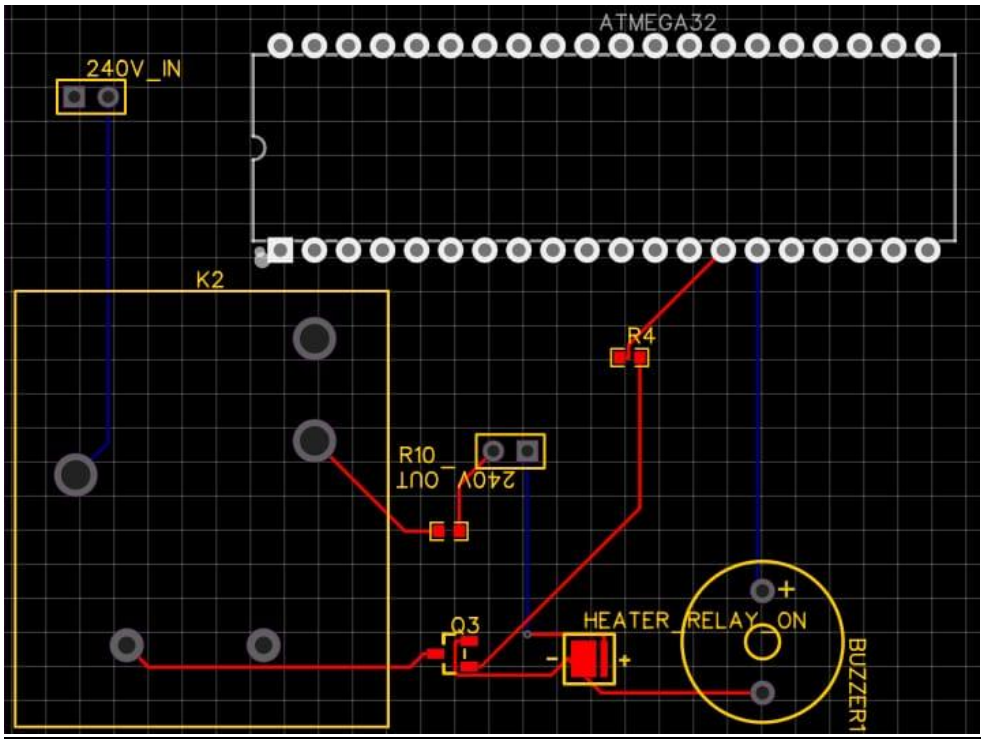
- It can be controlled by electronic circuits
- Produces a variety of pleasant sounds, and analog sound, intermittent sound.
- Pure timbre, not easily covered by noise.
- Has larger frequency ranges and SPL values,
- Produces a high resonant frequency
- Works under higher operating voltages and lower current requirements.

Component	Piezoelectric Buzzer
Type	Pin terminal type PS12 (PS1240P02BT)
Rated Voltage	~3 to ~250 V
Rated Current consumption	< 30 mA
Resonant frequencies	2 to 6 kHz
Operating temperature range	–10 to +70°C
Sound pressure	70dB(A)/10cm min
Pin Description	2 pins (Power pin, Ground Pin)

Circuit



PCB design for buzzer and heating coil



Code

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdbool.h>


int main(void)
{
       DDRD = 0xFF;

    while (1)
    {
              PORTD = 0x02;
              _delay_ms(5000);
              PORTD = 0x00;
              _delay_ms(5000);
    }
}
```

2. Heating coil

The heating coil is a electric device that is used to heat water or air. The coil is made from ceramic or metal and heated through electric current. It acts as a large resistor and as the electric current flows through it, it begins to heat up.

In our project we have to heat the sap of Kithul flower until the temperature set to specified temperature. Whole process of our project depends on the heating process. So, I decided to use an electric heating coil and tried to reduce user interaction by using automated way.

Technique and Specification

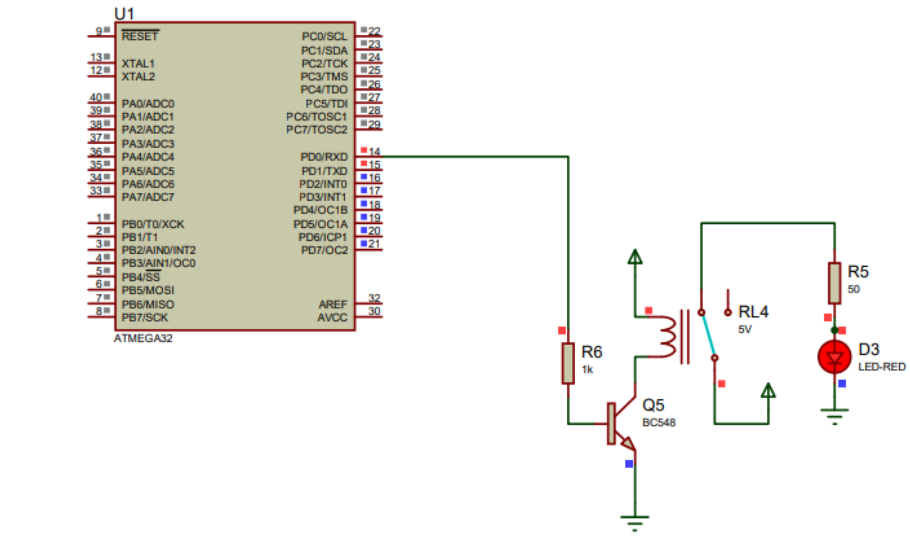
Heating coil is used to convert electric power into heat. When it comes to the technique of the Heating Coil,

Electronic circuits include resistors, which control how much current flow. Even a thin piece of wire will get hot if you force enough electricity through it. In an incandescent lamp, there's a very thin coil of wire called a filament that glows white hot.

Component	Heating Coil
Type	Round Coil Heater
Rated Voltage	230 to 440 V
Power (Watts)	50-10000W
Rated Current	≤60mA
Frequency	50, 60 Hz
Temperature	-5C - 350C

Circuit

To demonstrate the Heating Coil, I use a relay module.



Code

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdbool.h>


int main(void)
{
	DDRD = 0xFF;

    while (1)
    {
		PORTD = 0x03;
		_delay_ms(5000);
		PORTD = 0x00;
		_delay_ms(5000);


    }
}
```

2.Solenoid Valve

According to our project functions and requirements, we use three solenoid valves for specific purposes. Since the solenoid valves should have to work under high temperature, I choose to use **US Series High-Temperature Solenoid Valves** for the project.

Basically, there are three solenoid valves

- ✓ Valve 01: Filtering Valve

  The purpose of the Filtering Valve is to control sap from filtering part to container. In our project first, we have to pour sap to filter and measure the weight. Once the weight equals to 4kg the buzzer ring and Filtering Valve will open so that sap can be poured into container automatically. Filtering Valve is placed in filter.

- ✓ Valve 02: Treacle Valve

  The purpose of the Treacle Valve is to take out the Kithul Treacle from the container. Once the temperature of the mixture turns to treacle making temperature (predefined or set) the buzzer will ring so that user knows the heating and process is completed. Then user can take out the Kithul Treacle from the container by using key pad. Treacle Valve is placed in container.

- ✓ Valve 01: Jaggery Valve

  The purpose of the Jaggery Valve is to take out the Kithul Jaggery from the container. Once the temperature of the mixture turns to jaggery making temperature (predefined or set) the buzzer will ring so that user knows the heating and process is completed. Then user can take out the Kithul Jaggery from the container by using key pad.Jaggery Valve is placed in container.

Figure 18 *Solenoid Valve*

Technique and Specification

The Solenoid Valve is an electrically controlled valve. It consists of two main components; a solenoid and a valve body. It can be normally open or normally closed. In the de-energized state, a normally open valve is open and a normally closed valve is closed. The Solenoid Valve has an electromagnetically inductive coil, around an iron core at the center called the plunger

When current flows through the solenoid, the coil is energized and creates a magnetic field. Then creates a magnetic attraction with the plunger, moving it, and overcoming the spring force.

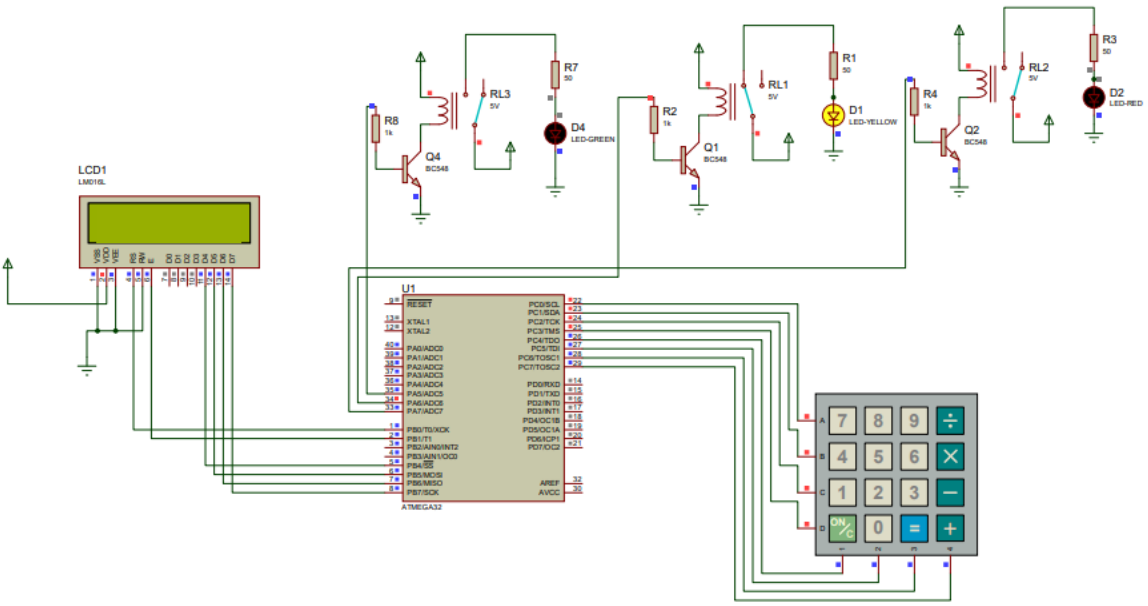
If the valve is normally closed, the plunger is lifted so that the seal opens the orifice and allows the flow. If the valve is normally open, the plunger moves downward so that the seal blocks the orifice and stops the flow. The shading ring prevents vibration and humming in AC coils.

Use the three most important operating principles direct-acting, indirect-acting, and semi-direct acting operation.

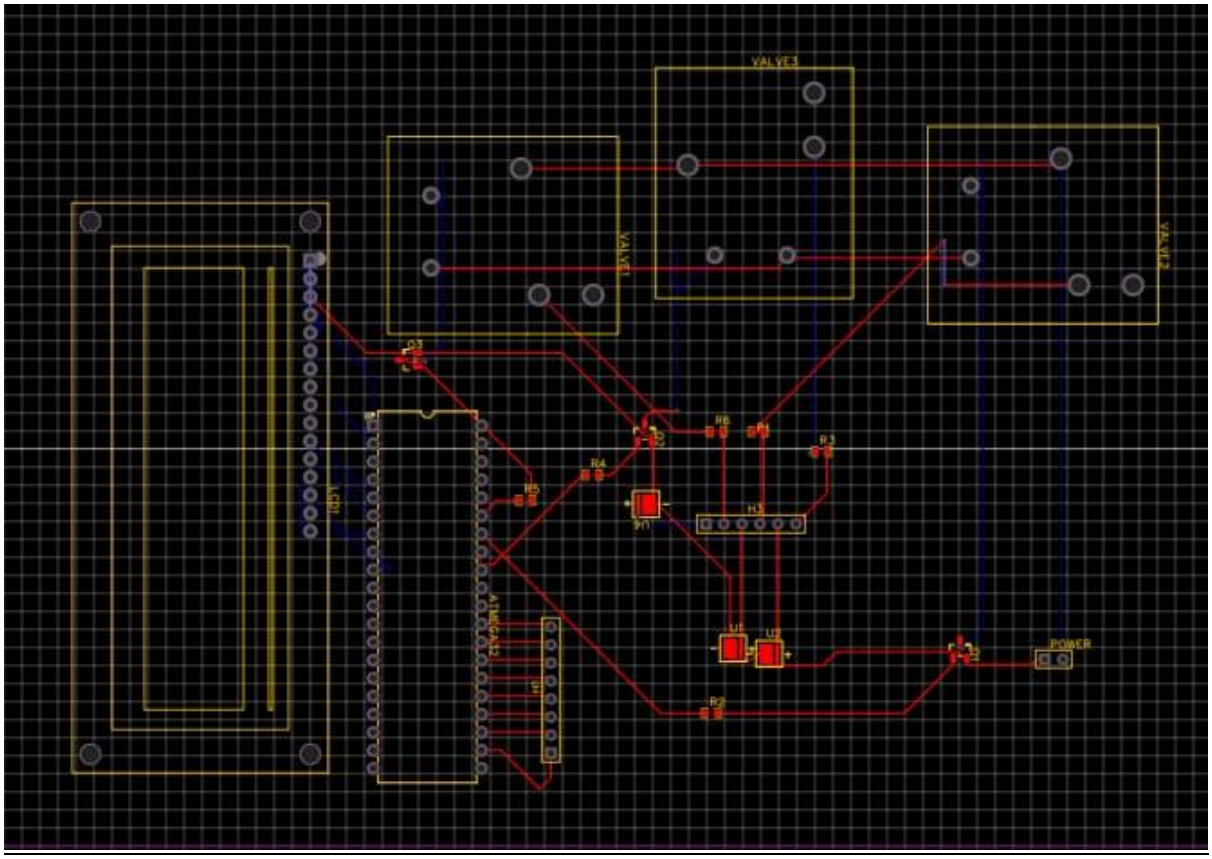
Component	Solenoid Valve
Type	US Series High-Temperature Solenoid Valves
Rated Voltage	12V DC
Operating Voltage	4 to 8V DC
Rated Current	≤60mA
Pressure Range	0.05 – 1.5 Mpa
Temperature	-5C - 350C

Circuit

When I design the circuit for the solenoid valves in proteus software, there were not a specific symbol for solenoid valve. So, to demonstrate the solenoid valve I use a relay module.



PCB design



Code

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include <avr/interrupt.h>


////////////////LCD display////////////////

#define LCD_Dir DDRB                                /* Define LCD data port direction */
#define LCD_Port PORTB                                      /* Define LCD data port */
#define RS PB0                                                  /* Define Register
Select (data reg./command reg.) signal pin */
#define EN PB1                                                  /* Define Enable signal
pin */

//////////////////////////define key pad//////////////////////
#define KEY_PRT PORTC
#define KEY_DDR DDRC
#define KEY_PIN PINC

unsigned char keypad[4][4] = {
       {'7', '4', '1', 'C'},
       {'8', '5', '2', '0'},
       {'9', '6', '3', '='},
       {'/', 'x', '-', '+'}};

       unsigned char colloc, rowloc;

       bool valve_1_state = false;
       bool valve_2_state = false;
       bool valve_3_state = false;

       char keyfind();

       void valve1();
       void valve2();
       void valve3();

       void LCD_Command( unsigned char cmnd )
       {
              LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
              LCD_Port &= ~ (1<<RS);                          /* RS=0, command reg. */
              LCD_Port |= (1<<EN);                    /* Enable pulse */
              _delay_us(1);
              LCD_Port &= ~ (1<<EN);

              _delay_us(200);

              LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
              LCD_Port |= (1<<EN);
              _delay_us(1);
              LCD_Port &= ~ (1<<EN);
              _delay_ms(2);
```

```
	}


	void LCD_Char( unsigned char data )
	{
		LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
		LCD_Port |= (1<<RS);			/* RS=1, data reg. */
		LCD_Port|= (1<<EN);
		_delay_us(1);
		LCD_Port &= ~ (1<<EN);

		_delay_us(200);

		LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
		LCD_Port |= (1<<EN);
		_delay_us(1);
		LCD_Port &= ~ (1<<EN);
		_delay_ms(2);
	}

	void LCD_Init (void)				/* LCD Initialize function */
	{
		LCD_Dir = 0xFF;					/* Make LCD
command port direction as o/p */
		_delay_ms(20);					/* LCD Power ON
delay always >15ms */

		LCD_Command(0x33);
		LCD_Command(0x32);			/* send for 4 bit
initialization of LCD  */
		LCD_Command(0x28);		/* Use 2 line and initialize 5*7
matrix in (4-bit mode)*/
		LCD_Command(0x0c);		/* Display on cursor off*/
		LCD_Command(0x06);		/* Increment cursor (shift cursor to
right)*/
		LCD_Command(0x01);		/* Clear display screen*/
		_delay_ms(2);
		LCD_Command (0x80);				/* Cursor 1st row 0th
position */
	}

	void LCD_String (char *str)			/* Send string to LCD function
*/
	{
		int i;
		for(i=0;str[i]!=0;i++)				/* Send each char of
string till the NULL */
		{
			LCD_Char (str[i]);
		}
	}

	void LCD_String_xy (char row, char pos, char *str)	/* Send string to LCD
with xy position */
	{
		if (row == 0 && pos<16)
		LCD_Command((pos & 0x0F)|0x80);		/* Command of first row and
required position<16 */
		else if (row == 1 && pos<16)
		LCD_Command((pos & 0x0F)|0xC0);		/* Command of first row and
required position<16 */
```

```
		LCD_String(str);					/* Call LCD string
function */
	}

	void LCD_Clear()
	{
		LCD_Command (0x01);				/* Clear display */
		_delay_ms(2);
		LCD_Command (0x80);				/* Cursor 1st row 0th
position */
	}

	int main(void)
	{
		DDRA = 0b11111111;
		LCD_Init();
		LCD_Command(0xc1);
		LCD_String("welcome");
		_delay_ms(1000);
		LCD_Clear();


		while (1)
		{
			char a = keyfind();
			LCD_Char(a);
			_delay_ms(1000);
			LCD_Clear();
			LCD_Clear();
			if (a == '1')
			{
				valve1();
			}
			else if (a == '2')
			{
				valve2();
			}
			if (a == '3')
			{
				valve3();
			}
		}
	}

	void valve1(){

		if (valve_1_state == false)
		{
			PORTA |= (1 << PINA5);
			valve_1_state = true;
		}
		else{
			PORTA &= ~(1 << PINA5);
			valve_1_state = false;
		}
	}

	void valve2(){

		if (valve_2_state == false)
		{
			PORTA |= (1 << PINA6);
```

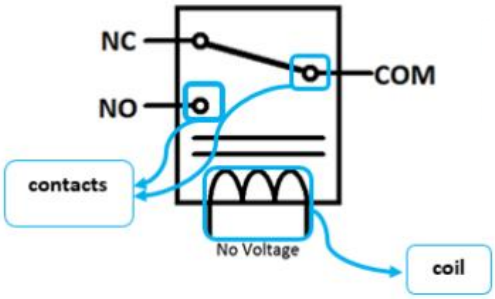
```
                valve_2_state = true;
        }
        else{
                PORTA &= ~(1 << PINA6);
                valve_2_state = false;
        }
}
        void valve3(){

                if (valve_3_state == false)
                {
                        PORTA |= (1 << PINA7);
                        valve_3_state = true;
                }
                else{
                        PORTA &= ~(1 << PINA7);
                        valve_3_state = false;
                }
        }

char keyfind()
{
        while (1)
        {
                KEY_DDR = 0xF0; /* set port direction as input-output */
                KEY_PRT = 0xFF;
                do
                {
                        KEY_PRT &= 0x0F; /* mask PORT for column read only */
                        asm("NOP");
                        colloc = (KEY_PIN & 0x0F); /* read status of column */
                } while (colloc != 0x0F);
                do
                {
                        do
                        {
                                _delay_ms(20);              /* 20ms key debounce
time */
                                colloc = (KEY_PIN & 0x0F); /* read status of column
*/
                                } while (colloc == 0x0F);      /* check for any key
press */
                                _delay_ms(40);                  /* 20 ms key
debounce time */
                                colloc = (KEY_PIN & 0x0F);
                        } while (colloc == 0x0F);
                        /* now check for rows */
                        KEY_PRT = 0xEF; /* check for pressed key in 1st row */
                        asm("NOP");
                        colloc = (KEY_PIN & 0x0F);
                        if (colloc != 0x0F)
                        {
                                rowloc = 0;
                                break;
                        }
                        KEY_PRT = 0xDF; /* check for pressed key in 2nd row */
                        asm("NOP");
                        colloc = (KEY_PIN & 0x0F);
                        if (colloc != 0x0F)
                        {
                                rowloc = 1;
                                break;
```

```
                }
                KEY_PRT = 0xBF; /* check for pressed key in 3rd row */
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if (colloc != 0x0F)
                {
                        rowloc = 2;
                        break;
                }
                KEY_PRT = 0x7F; /* check for pressed key in 4th row */
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if (colloc != 0x0F)
                {
                        rowloc = 3;
                        break;
                }
        }
        if (colloc == 0x0E)
        return (keypad[rowloc][0]);
        else if (colloc == 0x0D)
        return (keypad[rowloc][1]);
        else if (colloc == 0x0B)
        return (keypad[rowloc][2]);
        else
        return (keypad[rowloc][3]);
}
```

Relay Module

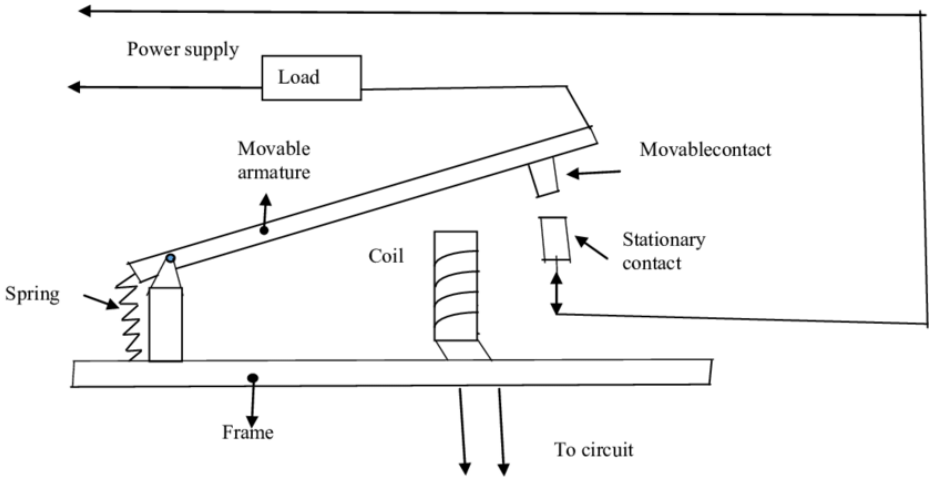
Relay module can be identified as an electrical switch. That operates electromagnet. That electromagnet can be activated by giving a separate low-power signal from a microcontroller and when it is activated, the electromagnet pulls to either open or close another circuit. Simply relays can control one electric circuit by using another circuit.

Pin diagram



Relays consists of 3 pins; normally open (NO), normally closed (NO), and a common pin. Apart from that coil pin is there.

Working Mechanism



According to above diagram, relay involves two circuit; energizing circuit and contact circuit. The coil is on energizing side. When the coil is energized by connecting a circuit to give power, then current flow through the coil and creates a magnetic field. Once the coil creates magnetic field, the movable armature attracts towards to the coil and then it completes the connection to contact circuit. Then the contact circuit activates.

So, using this mechanism I demonstrate the solenoid valves.

11.0 Table of Figures

12.0 References

[1] *Buzzer circuit problem in Proteus*. (n.d.). Stackexchange.Com. Retrieved November 30, 2021, from https://electronics.stackexchange.com/questions/519498/buzzer-circuit-problem-in-proteus

[2] *Jaggery making plant 9782099827*. (2021, May 31).

[3] *keypad interfacing with Atmega32*. (2014, September 27). Mechaterrain.Com. https://www.mechaterrain.com/keypad-interfacing-atmega32

[4] *Kithul Jaggery Cube Making Machine Plant with Low Price - Buy Jaggery Cube Making Machine, Kithul Jaggery, Jaggery Plant Product on Alibaba.Com*. (n.d.). Alibaba.Com. Retrieved November 30, 2021, from https://www.alibaba.com/product-detail/kithul-jaggery-cube-making-machine-plant_60766353109.html

[5] *Thermocouple Interfacing with AVR ATmega16/ATmega32*. (n.d.). Electronicwings.Com. Retrieved November 30, 2021, from https://www.electronicwings.com/avr-atmega/thermocouple-interfacing-with-atmega16-32

[6] (N.d.). Gov. In. Retrieved November 30, 2021, from https://aicrp.icar.gov.in/phet/wp-content/uploads/2017/02/Jaggery-machinery.pdf