

# Brewing Beer with the Portlet

Or basically using the Portlet to maintain a desired temperature

**Summary:** In this example we're showing how to use the Arduino to maintain a temperature. For example, when fermenting beer you can use the Arduino to maintain optimal fermenting temperature. This is NOT a tutorial on how to make beer. Sorry. The code for programming the Arduino is pasted in Appendix A

## Table of Contents

Overview of Fermenting Beer.....	2
Programming the Portlet.....	3
The Setup.....	6
Defining the temperature curve.....	7
Appendix A: Sample Arduino Code.....	8

# Overview of Fermenting Beer

The basic steps in making beer are:

## 1. Brewing the Beer

Malt extract and hops get boiled in water to sterilize the batch and extract all the goodness into the liquid.

## 2. Cooling and Fermenting

The mixture from step 1 is cooled to room temperature and transferred to a fermentation vessel with some additional water to achieve the desired batch size. Once cooled to an appropriate temperature the yeast is added to start the fermentation process. This is where Portlet comes in! Not only can Portlet help monitor your batch for a good time to pitch the yeast, but it can then help to keep the fermenting vessel at the desired temperature. That is what the rest of this tutorial will show.

## 3. Priming and Bottling

Once the brew is fully fermented, it is usually siphoned into another bottling container, or directly into bottles. A little extra sugar usually gets added to keep the yeast happy and create the carbonation.

## 4. Aging

After the beer has sat in the bottles for anywhere from 1-6 weeks it is ready to be refrigerated. During fermentation the yeast will have consumed the remaining sugar and carbonated the drink.

## 5. Drinking

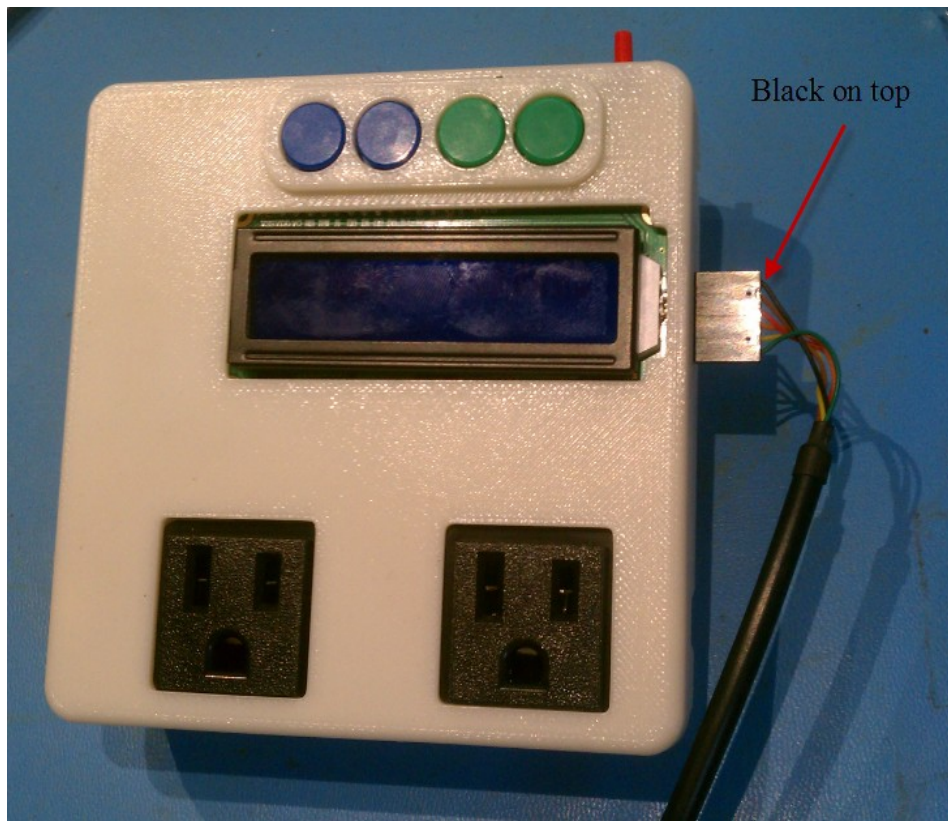
If you are reading this, we can be pretty sure you're familiar with this part.

Brewing beer can be a fun hobby! The internet has a lot of excellent resources and there are numerous books on the process. Do not get overwhelmed; it can be as simple as making elaborate tea, or you can make it as complicated and detailed as you would like.

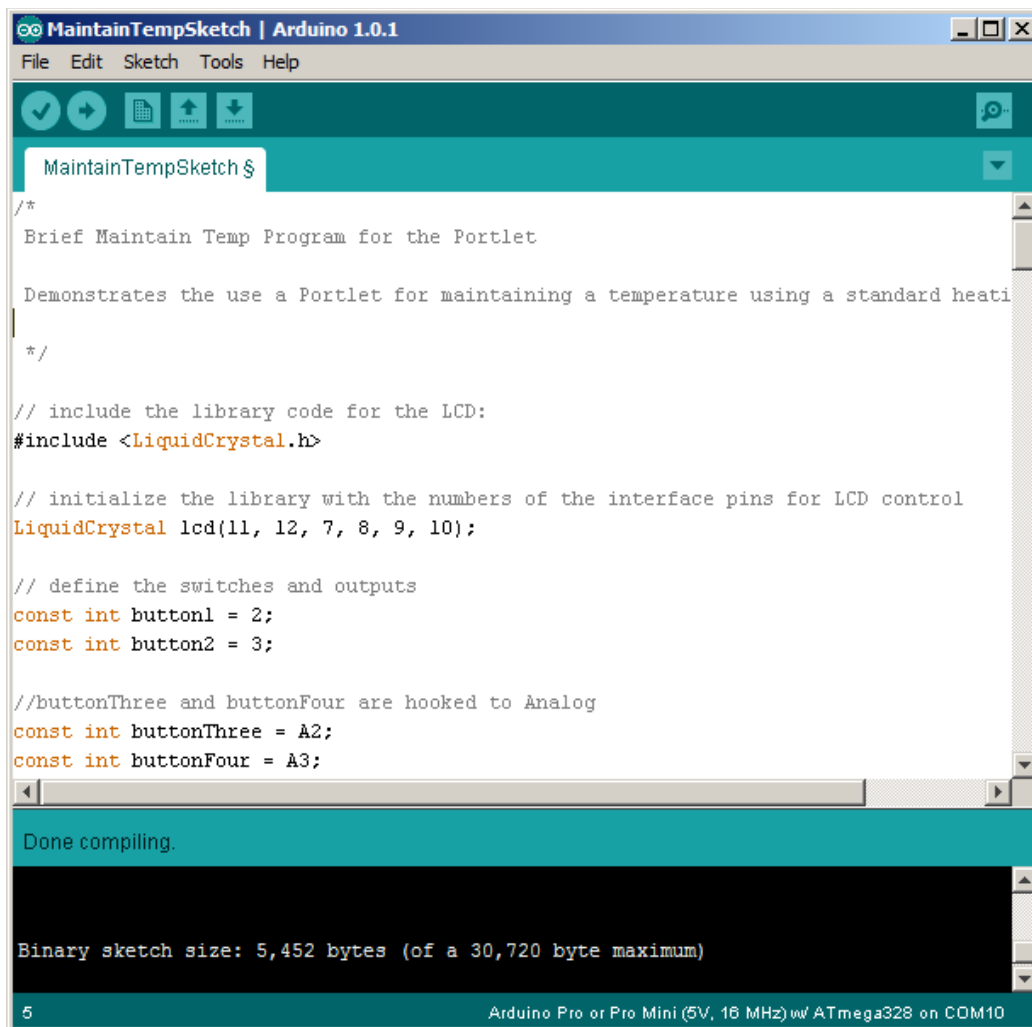
## Programming the Portlet

Before you start, it's best to get the Portlet all ready and programmed. The Portlet is based off the popular Arduino platform and uses the Arduino IDE for programming. Here we give a brief example of how to program the Arduino, so if you want to learn more you should head over to the excellent Arduino home page at <http://www.arduino.cc/>

First off, plug your programming cable into the programming interface of the Portlet. The interface is the six header pins coming off of the Arduino Pro Mini build into the device. There is an access port in the covers on the right side of the Portlet. If you are using the Sparkfun 5V programming cable, the black wire goes on top for the correct orientation.



Open up the Arduino IDE and copy and paste the Maintain Temp Program attached into the IDE.



```
MaintainTempSketch $
/*
  Brief Maintain Temp Program for the Portlet

  Demonstrates the use a Portlet for maintaining a temperature using a standard heati
*/

// include the library code for the LCD:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins for LCD control
LiquidCrystal lcd(11, 12, 7, 8, 9, 10);

// define the switches and outputs
const int button1 = 2;
const int button2 = 3;

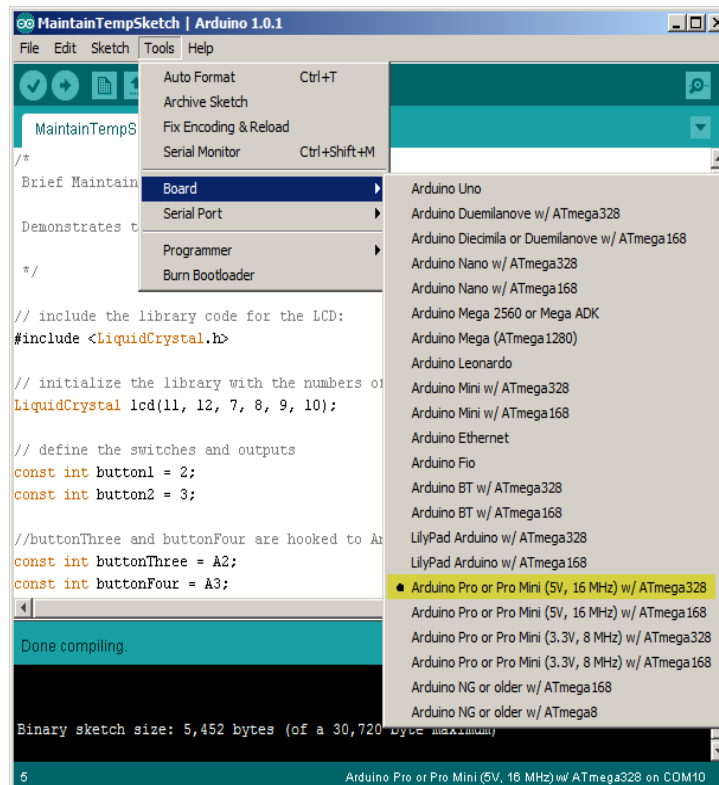
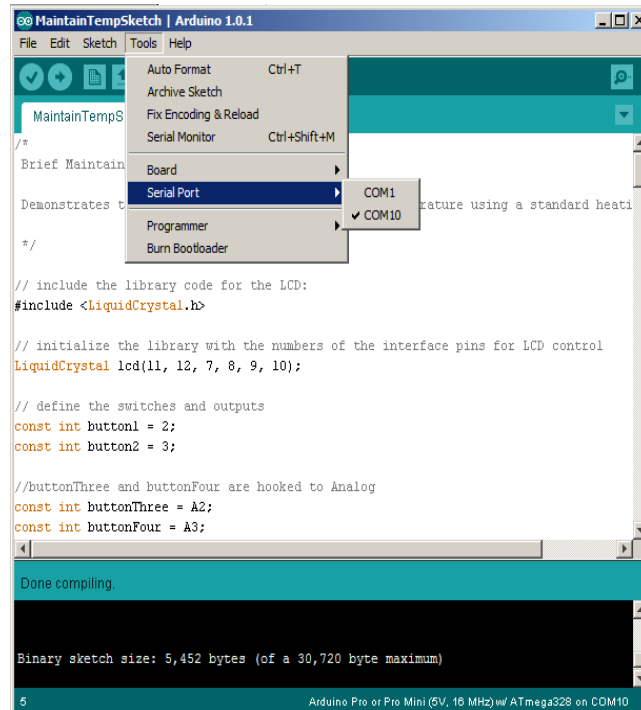
//buttonThree and buttonFour are hooked to Analog
const int buttonThree = A2;
const int buttonFour = A3;
```

Done compiling.

Binary sketch size: 5,452 bytes (of a 30,720 byte maximum)

5 Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328 on COM10

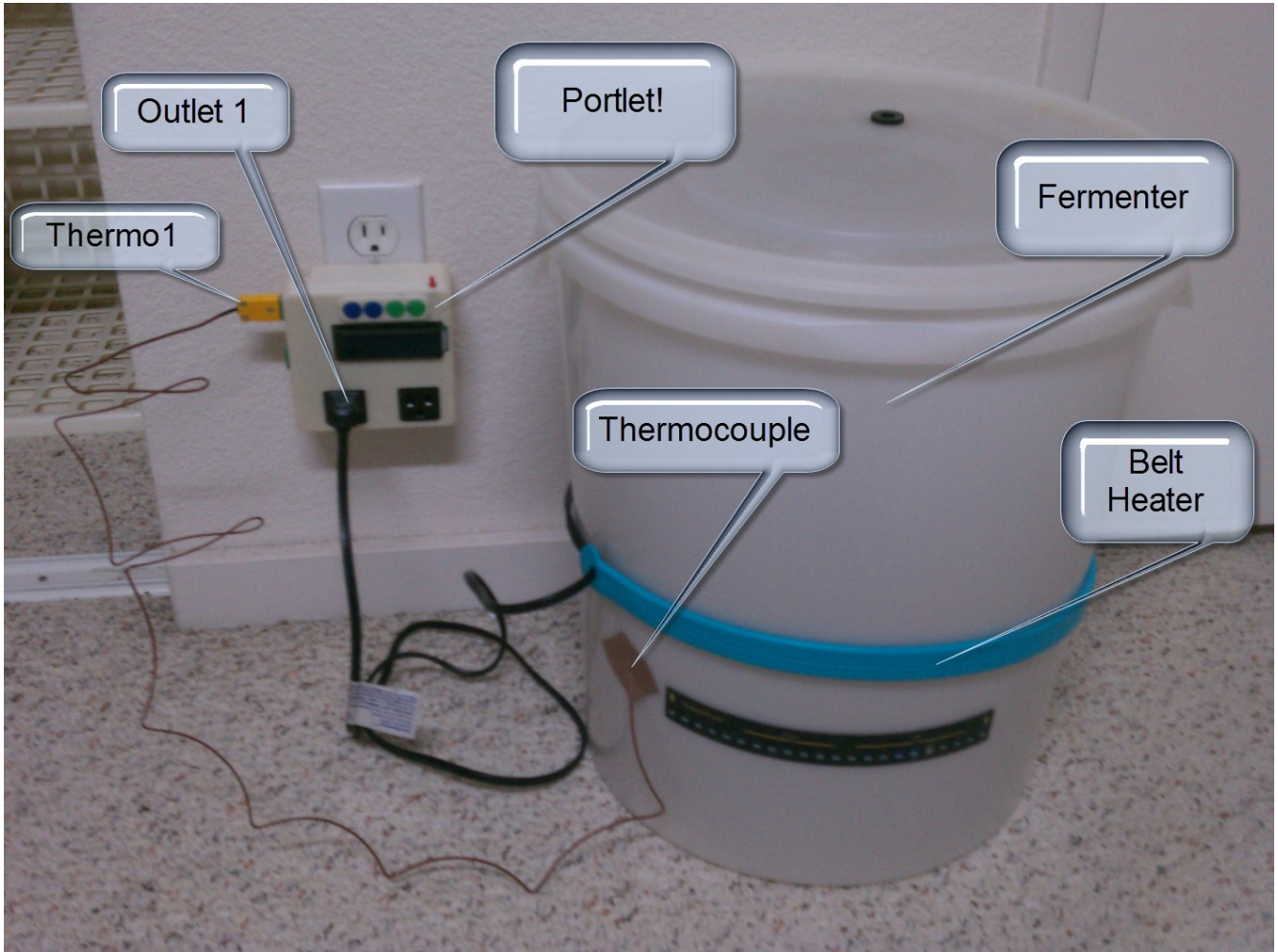
Make sure you have the correct serial port selected for your interface and for the board choose the “Arduino Pro or Pro Mini (5V, 16MHz) w/ATmega328”



## The Setup

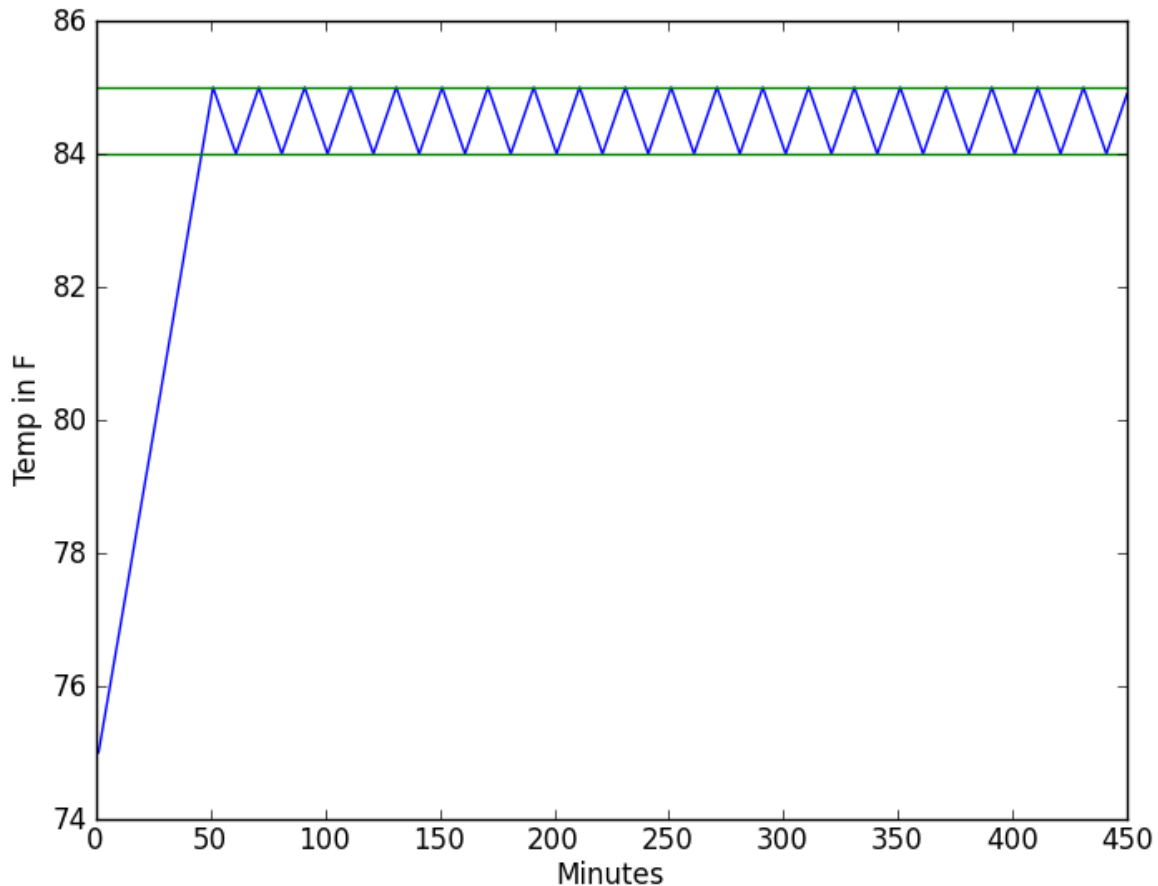
Here you see the fermenter with the belt heater and Portlet all hooked up. The belt heater by itself tends to heat the batch too high for good fermenting. So the Portlet monitors the fermenter and makes sure that the belt heater is on only periodically to keep the batch at the right temperature.

The program supplied below expects that you are using Outlet 1 and Thermo 1. Make sure you have the belt heater and the thermocouple plugged into the right sockets!



## Defining the temperature curve

If we are to graph this process, the basic temperature profile would look something like:



The slopes shown above are made up just to give you an idea of how this is working. Those aren't the actual rise and fall rates of the temperature. That depends on your setup; how beefy the belt heater is, how large the batch is, etc.

Above you can see the temperature start at ambient and then ramp up to 85°F (set in the program). After that you can see it jumps back and forth between 84 and 85. This is just how the program is set up. Because of the large thermal mass (5 gallons of beer in the making) the temperature moves very slowly so this is an adequate means of maintaining a temperature.

It is possible to set up real PID control with the Portlet, but it is overkill in the case of this example.

## Appendix A: Sample Arduino Code

```
/*  
Brief Maintain Temp Program for the Portlet  
  
Demonstrates the use a Portlet for maintaining a temperature using a standard heating device.  
  
*/  
  
// include the library code for the LCD:  
#include <LiquidCrystal.h>  
  
// initialize the library with the numbers of the interface pins for LCD control  
LiquidCrystal lcd(11, 12, 7, 8, 9, 10);  
  
// define the switches and outputs  
const int button1 = 2;  
const int button2 = 3;  
  
//buttonThree and buttonFour are hooked to Analog  
const int buttonThree = A2;  
const int buttonFour = A3;  
  
const int thermo1 = A0;  
const int thermo2 = A1;  
  
const int outlet1 = 6;  
const int outlet2 = 5;  
  
//the LCD backlight is used as feedback to the user  
const int backlight = 13;  
  
void setup() {
```



```

// set up the LCD's number of columns and rows:
lcd.begin(16, 2);

//define buttons as inputs and outlets as outputs
pinMode(button1, INPUT);
pinMode(button2, INPUT);

pinMode(outlet1, OUTPUT);
pinMode(outlet2, OUTPUT);

//enable serial for troubleshooting
Serial.begin(9600);

}

int convert_degrees(int analog_value){
  int temperature = 0;

  //NOTE: IF YOU WANT TO FINE TUNE YOUR TEMPERATURE READOUTS
  //THIS IS THE PLACE TO TO DO IT!
  int offset = 0;

  //the circuit changes 10mV/degree Celsius (hopefully)
  //Analog inputs measure 0 to 5V
  //0 to 5V is a range of 0 to 500 degrees Celsius

  //UNCOMMENT THIS LINE FOR CELSIUS
  //temperature = map(analog_value, 0, 1023, 0, 500) + offset;

  //0 to 500 degrees Celsius is 32 to 932 degrees Fahrenheit
  //UNCOMMENT THIS LINE FOR FARENHEIT
  temperature = map(analog_value, 0, 1023, 32, 932) + offset;

```

```
return temperature;
}
```

```
//turn the outlet on until target temperature is reached
```

```
void heat_up(int target_temp){
  //while temp is less than target, keep outlet on
  while(target_temp > convert_degrees(analogRead(thermo1))){
    digitalWrite(outlet1, HIGH);
    //reset LCD and print current operation
    lcd.clear();
    lcd.home();
    lcd.print("Heating to ");
    lcd.print(target_temp);
    //set cursor for printing temp
    lcd.setCursor(0,1);
    //print current temperature
    lcd.print("Temp:");
    //clear the old temp from the screen
    lcd.setCursor(8, 1);
    lcd.print("  ");
    lcd.setCursor(8, 1);
    //print current temp
    lcd.print(convert_degrees(analogRead(thermo1)));
    //pause for a bit so the screen doesn't flicker
    delay(200);
  }
  //turn outlet off before exiting
  digitalWrite(outlet1, LOW);
}
```

```
//turn the outlet off until it cools to target temperature
```

```
void cool_down(int target_temp){
  //make sure outlet is off
  digitalWrite(outlet1, LOW);
}
```

```

//wait for temperature to drop below target
while(target_temp < convert_degrees(analogRead(thermo1))){
    //clear lcd and print current operation
    lcd.clear();
    lcd.home();
    lcd.print("Cooling to: ");
    lcd.print(target_temp);
    lcd.setCursor(0, 1);
    //print current temperature
    lcd.print("Temp:");
    //clear the old temp from the screen
    lcd.setCursor(8, 1);
    lcd.print("  ");
    lcd.setCursor(8, 1);
    //print current temp
    lcd.print(convert_degrees(analogRead(thermo1)));
    //pause for a bit so the screen doesn't flicker
    delay(200);
}
}

```

```

//signalling the user to pitch the yeast
void pitch_yeast(){
    //clear lcd and print current operation
    lcd.clear();
    lcd.home();
    lcd.print("Pitch Yeast");
    lcd.setCursor(0, 1);
    lcd.print("1 to continue");
    //flash the backlight to get attention
    int time_elapsed = millis();
    while(digitalRead(button1) == LOW){
        if((millis()-time_elapsed) > 1000){

```

```

    digitalWrite(backlight, HIGH);
}
if ((millis()-time_elapsed) > 2000){
    digitalWrite(backlight, LOW);
    time_elapsed = millis();
}
}
}

```

//routine to maintain the temperature at specified temp and for duration time

```

void maintain_temp(int target_temp, int time_in_min){
    //only check the temperature once per minute
    int num_loops = time_in_min;
    for(int x = 0; x < num_loops; x++){
        if (convert_degrees(analogRead(thermo1)) < target_temp){
            digitalWrite(outlet1, HIGH);
        }
        else {
            digitalWrite(outlet1, LOW);
        }
        //clear lcd and print current operation
        lcd.clear();
        lcd.home();
        lcd.print("Maintain temp");
        lcd.setCursor(0, 1);
        //display number of minutes
        lcd.print("Num min: ");
        lcd.print(x);
        lcd.print(" ");
        //pause 1 minute each cycle
        delay(60000);
    }
    //turn outlet off before exiting
    digitalWrite(outlet1, LOW);
}

```

```
}
```

```
void loop() {  
  // prepare LCD  
  lcd.clear();  
  lcd.setCursor(0, 0);  
  lcd.print("Press 1 ");  
  lcd.setCursor(0, 1);  
  lcd.print("to Start ");  
  
  //do nothing loop until a button is pressed  
  while(digitalRead(button1) == LOW);  
  
  //wait for batch to reach room temperature  
  cool_down(80);  
  //notify the user and tell them it's time to pitch the yeast  
  pitch_yeast();  
  //heat to 85 degrees  
  heat_up(85);  
  //culture the yeast at 100 degrees for 1 week  
  //remember to input the time in minutes  
  //One week is about 10080 minutes  
  maintain_temp(85, 10080);  
  
  //Signal success  
  lcd.clear();  
  lcd.setCursor(0, 0);  
  lcd.print("ALL DONE!");  
  lcd.setCursor(0, 1);  
  lcd.print("Refrigerate");  
  
  //either press button1 or reset to do it again
```

```
while(digitalRead(button1) == LOW);  
}
```