# Creating your own sous vide machine

## What is Sous Vide

If you don’t know what sous vide is you should read through this Wikipedia article <http://en.wikipedia.org/wiki/Sous-vide>

This system uses an Arduino microcontroller to implement a PID controller to precisely regulate the temperature of a water bath used for Sous Vide cooking. A PID controller allows the system to adapt and accurately control the temperature of virtually any configuration of heater and water bath. You can learn more about PID controllers here (<http://en.wikipedia.org/wiki/PID_controller>).

## Things you will need

* Arduino
* Temperature Probe
* Power Unit
* Heater
* User Interface

## Arduino

This system was developed using an Arduino Uno. I recommend you use an Uno for this project, however the system could probably be adapted to a Arduino MEGA without much effort. The system would require several major changes to work on a Duemillanova or older Arduino.

## Temperature Probe

The system is designed using an LM35 Precision Centigrade Temperature Sensor to measure temperature. This sensor allows the system to measure temperatures between the freezing and boiling point of water. The LM35 gives accurate temperature readings and is much cheaper and easier to implement than a traditional k-type probe system. The downside of the LM35 is that the sensor must be immersed in water.

### Building the temperature probe

|  |  |
| --- | --- |
| 1 | LM35 temperature sensor |
| 20ft | Stranded wire with insulation rated to at least 100° C |
| 1 ft | 1/8” Heat shrink tubing to protected sensor leads |
| 2 in | 1/4" Heat shrink tubing to protect wires |
| 2 in | ¼” OD Copper tubing. Sold at hardware store for connecting to refrigerator water supply. Normally sold in 10ft lengths. |
| 1 tube | Epoxy |
|  |  |

Step 1 — Solder wire to each of the 3 leads to the LM35. Before soldering slip a length of 1/8” heat shrink tubing over each of the wires. Strip the insulation off 1/2" inch from the wire. Solder each wire to each of the leads, making sure each wire is well bonded to the lead. Slip the heat shrink tubing over the solder joints and apply heat to strengthen and insulate the connections. Strip 3/8” of insulation off the other end of the wire, twist the wire, and tin the ends for connection to the Arduino. Mark the 3 wires so you will know which is which when the sensor is sealed.

Step 2 — Prepare copper tube to protect the sensor. Cut a 2” piece of the tube. It is best to use a tube cutter to make the cut. A saw could be used if you don’t have a tube cutter. The sensor will just barely fit in the tubing. You may need to use a tool to stretch open the ends that you cut to make it fit. There is a special flaring tool made for this purpose but if you don’t have access to one a phillips screwdriver works well.

Step 3 — Epoxy the sensor into the copper tubing. The Epoxy will seal the sensor into the tubing keeping the system water tight and protecting the system from physical damage. Make sure the end of the sensor is at least 1/4" from the end of the tubing. Fill both ends of the tubing with epoxy. Moving the sensor back and forth through the tube can help make sure you fill the tube and get all the air bubbles out. Set the probe aside for at least 24 hours to harden.

Step 4 — Place a piece of 1/4" heat shrink tubing over the end of the copper tubing and shrink the tubing to protect the wires coming out of the tubing from being damaged.

Congratulations the temperature probe is complete.

### Building the power module

|  |  |
| --- | --- |
| 1 | Solid State Relay (SSR) |
| 1 | Extension Cord |
| 1 | Weather resistant electrical 1 gang box |
| 1 | Cover for electrical box |
| 2 | PVC crimp connectors |

The power module allows the PID controller to on and off power to the heater to carefully regulate the temperature of the system. The system uses a Solid State Relay (SSR) to cycle the power on and off to the heater every 4 seconds. How long the power is applied to the heater is controlled by the PID controller.

The power module is basically an extension cord that is cut in half with the SSR spliced in an electrical box in the middle. When the extension cord is plugged into the wall and the heater is plugged into the extension cord. A very small DC voltage from the Arduino can switch power on and off to the heater.

#### How to find a SSR.

I found the easiest way to get an SSR is to buy one off eBay. I got one really cheap sent from China, but it took about 6 weeks to get here. The SSR must trigger off 5V DC, must be able to switch at least 10A but more will not hurt anything. You do not need a more expensive ‘fast’ SSR, what you are looking for is a zero cross trigger SSR which is the most common and cheapest type. This will ensure that the power will only turn on or off to the output when the AC voltage on the line is 0V which will prevent any complications with slightly inductive loads.

#### Construction

1. Cut extension cable in half at a point several feet away from the female plug end. Strip back outer insulation several inches on each side. UL certified extension cords will have 3 insulated wires white (HOT), black (NEUTRAL), and green (GROUND).
2. Mount SSR in side of electrical box. You will need to drill two holes on the side of the electrical box to securely bolt the SSR to the box. Securing the SSR is essential to protect the system from physical damage and to properly sink heat from the SSR into the electrical box.
3. Mount the two ends of the extension cable through the crimp connectors on either side of the electrical box.
4. Inside the electrical box connect the two green wires to the grounding screw inside the electrical box. Connect both white wires together with a wire not, verify that none of the conductors are exposed. Connect both black wires to the two AC terminals on the SSR.
5. Attach two wires to the DC side of the SSR, run the wires outside of the electrical box. Make sure that these wires are long enough to reach the Arduino controller. Mark the wires to indicate which wire connects to the + and – connectors.
6. Optional — attach an LED with a 1k ohm resistor to the SSR + and – terminals. This will indicate when the power module is delivering power.

## How to choose a heater

The heater at a minimum needs to hold water and use electricity to heat it up. I used a crock-pot to but many things could work. Make sure that any slow cooker you choose has no controls, or a simple mechanical switch. If the slow cooker has a digital controller the power module won’t be able to switch power off and on.

A slow cooker works and is a cheap and easy way to create a water bath, but it has some draw backs. Slow cookers are designed to cook slowly, which means that they have very weak heating elements and are really slow to heat the contents. Most slow cookers have a large heavy ceramic pot that absorbs lots of latent heat and transfers heat to the water bath very slowly and insulates the water bath very well. The combination of weak heating element, large heat in the ceramic pot, and lots of insulation makes it very difficult for the controller to precisely control the temperature.

Slow cooker base machines will be slow to heat and will be hard to ‘tune’ to get the system to accurately reach and hold the set temperature without much overshoot. In practice I was able to tune my crock-pot based system to get it very accurate but it took quite some time.

Some people swear by using a rice cooker as the heater, which seems a logical choice, they have much more powerful heaters and have less insulation which should make the systems with a rice cooker easier to tune.

I also considered, but haven’t tried, using an electric hot plate along with an ordinary large pot as a heater.

Some people making a very large water bath use very high power immersion water heaters.

## User Interface Choices

### Headless USB only interface

The system is designed to work using a computer connected to the Arduino by a USB cable to control the system. Commands can be sent to the system using the Arduino Serial Monitor. The following commands are accepted (SetTemp(142), SetP(10), SetI(5), SetD(10)). The system will send debug information to the computer every 4 seconds. See the source code for more information about this mode.

### Color LCD Screen Shield

The system is designed to work with a color LCD shield provided by SparkFun (<http://www.sparkfun.com/products/9363>). The shield will allow the user to change all the settings of the system and monitor the state of the system. This is really good way to go, but is obviously a more expensive way to go.

### Creating your own user interface

The system is designed to make it easy for anyone to step in and develop their user interface to the system. Any new user interface system needs to implement the UISystem abstract class and be assigned as the View to the Server class. See the code for more information.

## Connecting the whole thing together

The + connection to the SSR should must be attached to pin 10 of the Arduino and the – connection should be attached to the GND terminal nearby.

The temperature sensor needs 3 connections; 5V from the 5V terminal on the Arduino board, GND, and connect the output pin from the temperature sensor to PIN A5 on the Arduino board.

One needs to verify that the system is configured to use the appropriate view before compiling the system and uploading it to your Arduino. It is best to start with and tune the system with the Arduino connected to a computer with a serial monitor tuned on. The debug output send to the serial bus is invaluable when tuning the system and verify that the system is set up correctly.

## Tuning the system

The goal of a PID controller is to get the system to set temperature quickly, without overshoot (getting too hot), make the temperature of the system stable to the set point, and allow the system to adapt to changes in the system such as adding food to cook without breaking the stability of the system.

This PID controller system only has control over the heating element and no way to cool the system other than natural cooling of the system to room temperature, heating the system will naturally happen faster than cooling the system.

For the system to be as accurate as possible the three PID parameters (called Kp, Ki, and Kd) need to be set to the proper value for the system. Many attributes of the system affect what the ideal parameters of the system including: mass, insulation, heating wattage, ambient temperature, placement of the heater and temperature probe, etc…

How to tune the system is an art that is beyond the scope of this document. A basic method can be found on Wikipedia (<http://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method>). The basic goal is to get the system to heat to the set point, overshoot the set temperature by very little or no amount, and then stabilize at the set temperature with little or no fluctuation.

Make sure to read this Wikipedia article (<http://en.wikipedia.org/wiki/PID_controller>).

For my Crock-pot I discovered the best settings were (Kp:45, Ki:0.035, Kd:2100).