Programmable Temperature Controller + Hot Plate

by **BrittLiv** on November 25, 2013

Table of Contents

Author:BrittLiv Find out more about me

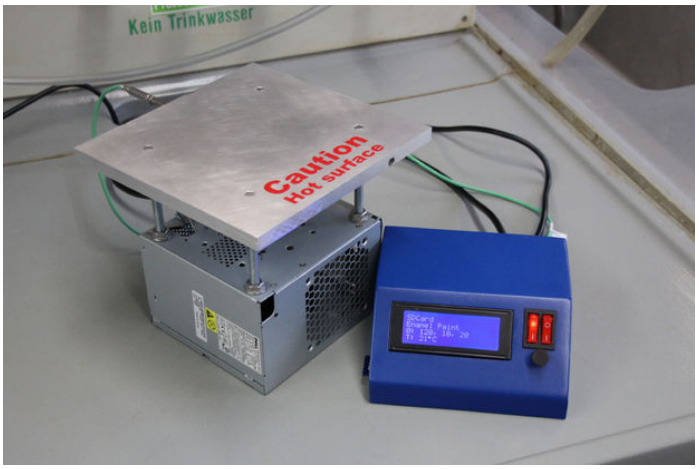
I'm a Chemical and Biological Engineering student especially interested in Computational Fluid Dynamics. To balance all the theoretical work I like to make stuff in my free time.

Intro: Programmable Temperature Controller + Hot Plate

Heating things up is one of the most performed tasks in a lab. Quite a lot of times it is not enough to simply hold something at a certain temperature, but the rate at which something is heated and for how long is just as important. Especially when you try to develop catalysts for chemical processes, the temperature program and exact temperature control is crucial and you probably do not want to stay in the lab for 16 hours to manually adapt your temperature program. Unfortunately, programmable temperature controllers that can automate processes are really expensive. So I decided to build a highly customizable controller that is able to run temperature ramps and read multiple different temperature programs from a SD card. It also provides a logging function on the SD-Card that allows you to evaluate the resulting temperature profile after running a program.

It is a great hack for your heating devices, since it can be easily connected to almost any heating apparatus you can think of, as long as it allows you to also connect a thermocouple. So If you have ever thought about building the perfect electric kiln (there are multiple really good explanations online) or hot plate (take a look at the steps 6 and 7), now is your time.

Should you like this ible I would really appreciate a vote. You can do so by clicking the orange flag in the top right corner.



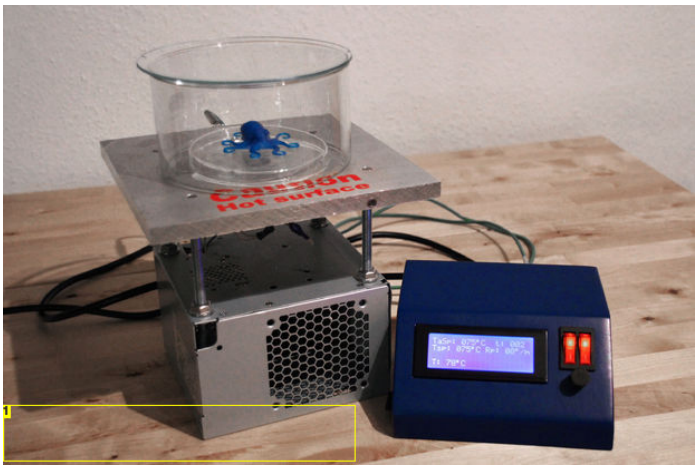


Image Notes
1. Smoothing PLA with THF

Step 1: Things you need

Programmable temperature controller:

Electronics:

- Solid state relays (5V control voltage, 16 A load current)
- LCD (e.g. on amazon.com)
- SD card board
- MAX 6675 controller board (e.g. on olimex.com)
- Atmega 328 chip & socket
- 5 x 10 kâ„¦ Resistors
- 4 x 1 kâ„¦ Resistors
- 4 x 560 â„¦ Resistors
- 1 µF Capacitor
- 100 pF Capacitor
- 2 x 22pF Capacitor
- 16 MHz quartz oscillator
- LM7805 5V linear voltage regulator
- Rotary encoder
- Mechanical 110 V switches
- 10 A fuse and fuse holder

For the casing:

- 4 mm ply wood
- Wood glue
- Laser cutter
- Primer
- Paint

Hot plate:

- Metal case as support (e.g. an old computer power supply)
- Small plastic case

- Aluminum plate 20x20x1 cm
- 2 Cartridge Heaters (1/4'' or 6 mm) (e.g. on amazon.com or from China on ebay.com)
- Type K thermocouple (1/4'' or 6 mm)
- Steel thread rod (¼ '' or 3/8'') and nuts
- Locking screws
- Lead
- Copper paste

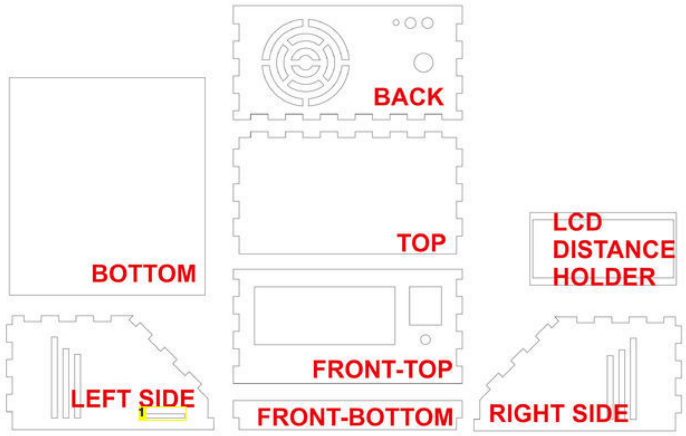
Tools

- Long (ca. 30 cm) drill for the heating cartridge and thermocouple
- Tap & die for Locking screw and steel thread rod
- Drill press

Step 2: Building the case part 1

You can of course use any casing you want, but I decided to laser cut a custom one. I uploaded the files to this step (the three different file types all include the same design. I just wanted to offer you options depending on which file type works best for you). You can see in the first image where every piece goes. Start by glueing the LCD distance holder to the back of the front-top panel. While the wood glue sets, attach the side panels to the back and the top panel. A corner clamp is certainly helpful to do so. Than add the front-bottom and front-top piece.

In case you are wondering whether wood is a good material choice for the casing, let me reduce your fears. Its autoignition temperature is about 572°F [1]. Since I have added a thermal switch to the hot plate that shuts it off at 338°F, you can be very sure that it is safe. Even after more than a thousand days at a constant temperature of 300°F, studies showed that wood would not self ignite [2]. Furthermore air is know to be an extremely good thermal insulator. If you are planning on running the temperature controller with a apparatus that heats up the surrounding air to more than 500°F: **DON'T** ! Even if you are building a kiln the outside shouldn't be getting that hot.



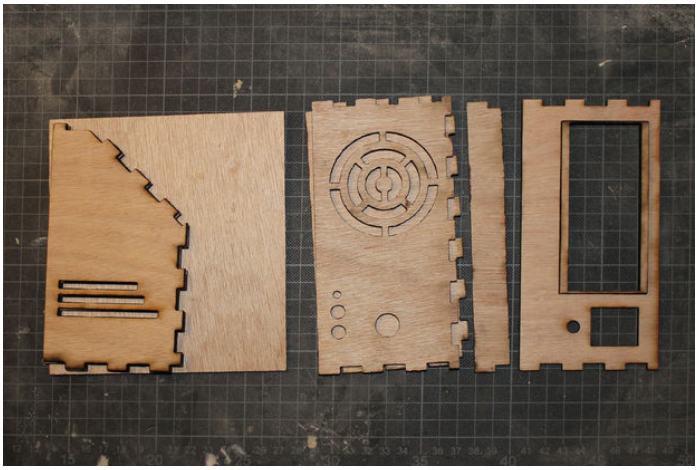
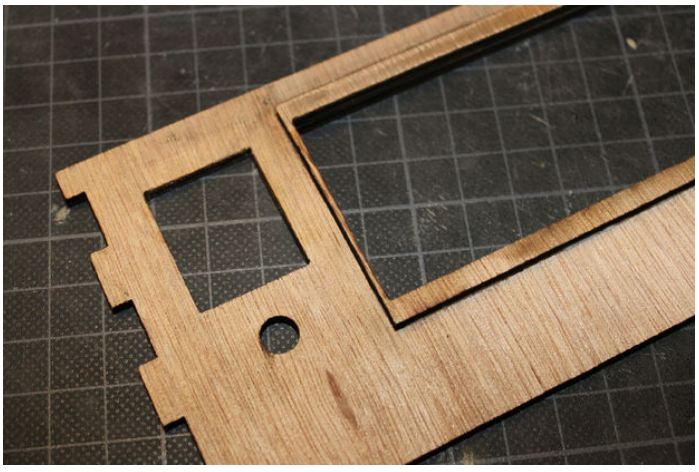
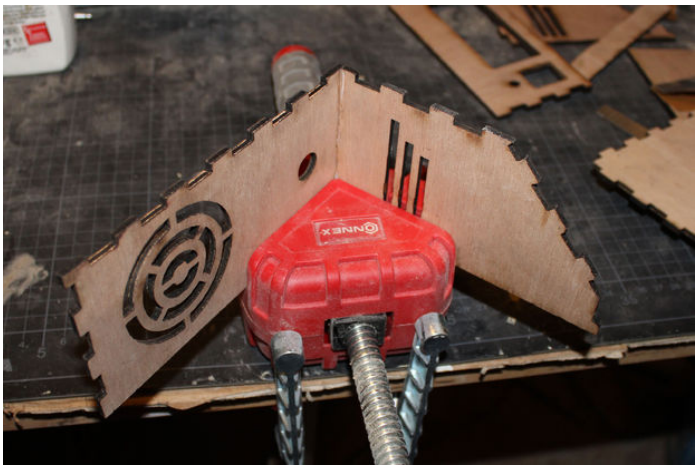
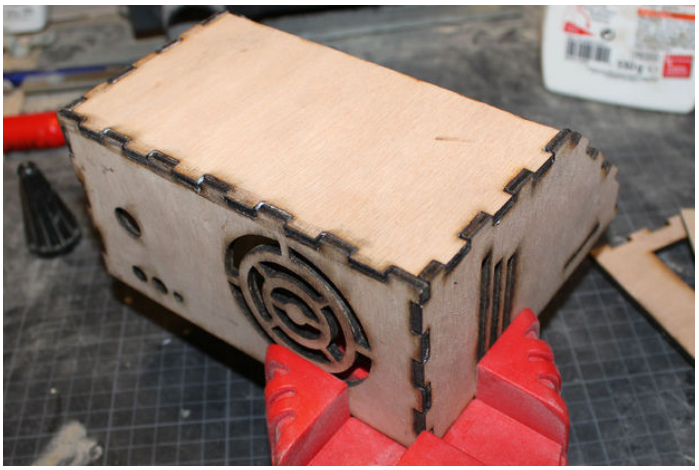
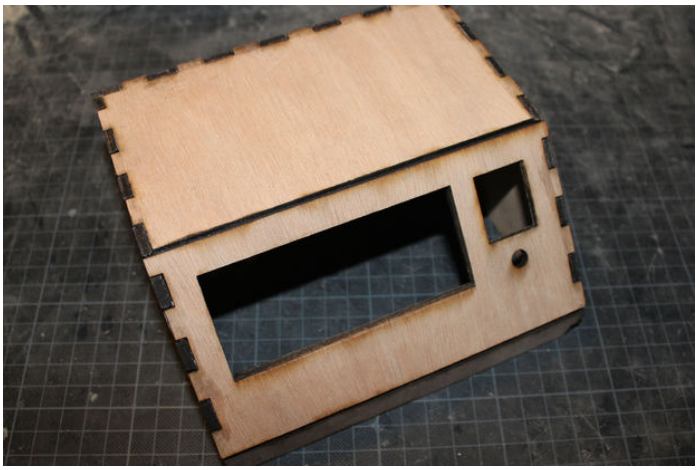


Image Notes
1. slot for the sd card









File Downloads

case.ai (18 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'case.ai']

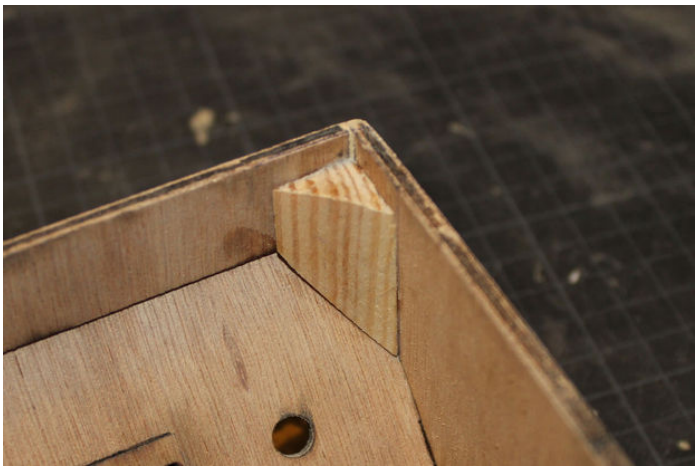
case.cdr (1 MB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'case.cdr']

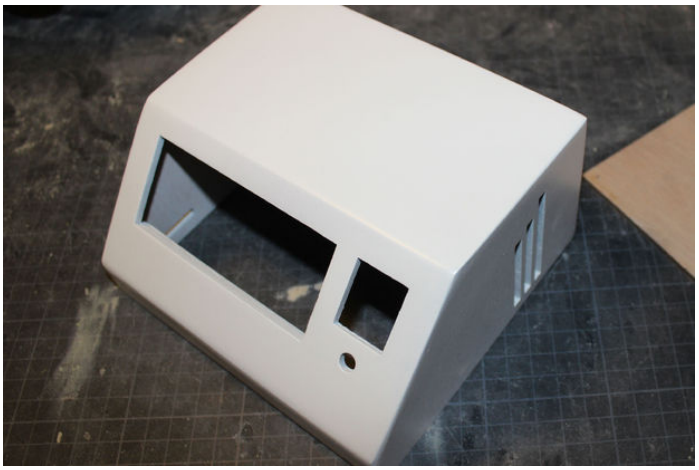
case.eps (2 MB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'case.eps']

Step 3: Building the case part 2

Since I am not a big fan of the burned edges the laser cutter leaves, I decided to get rid of them by filling the gaps with wood filler and sanding them down. You can see the result in the first picture.

I used a triangular shaped strip of wood to strengthen the connections. They will also be used to attach the bottom later on. Cut it to fit the corners, as shown in the pictures.
If you want you can prime and spray paint it, but that's up to you.

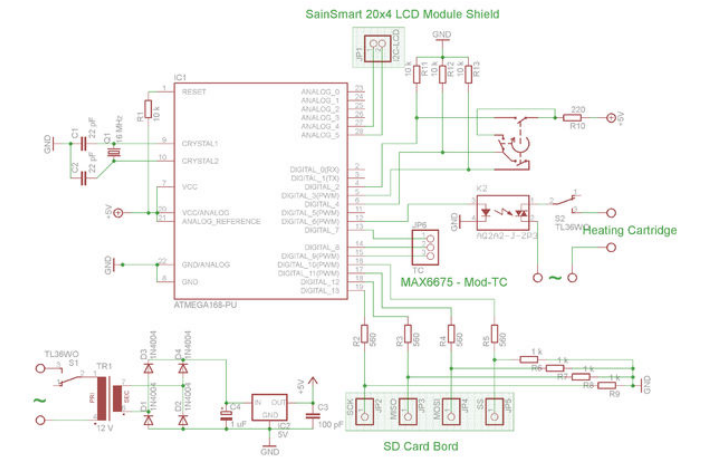


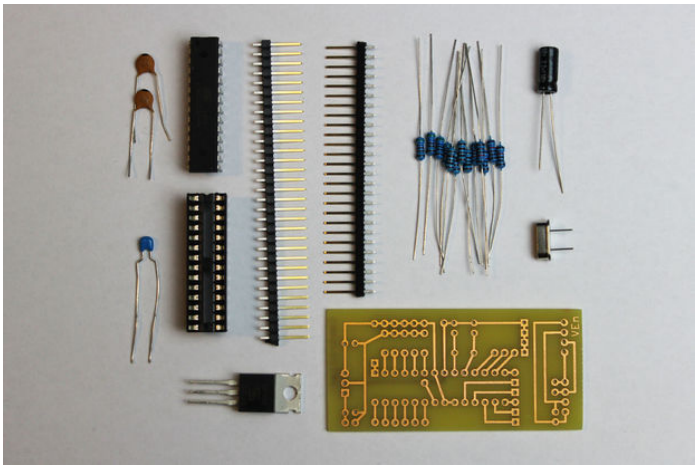


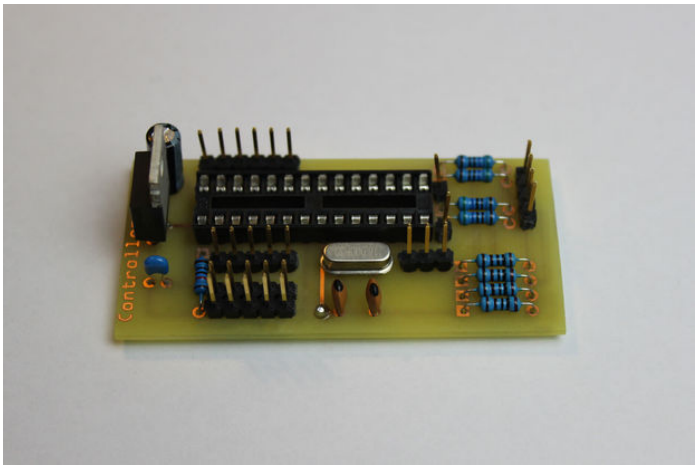
Step 4: Electronics

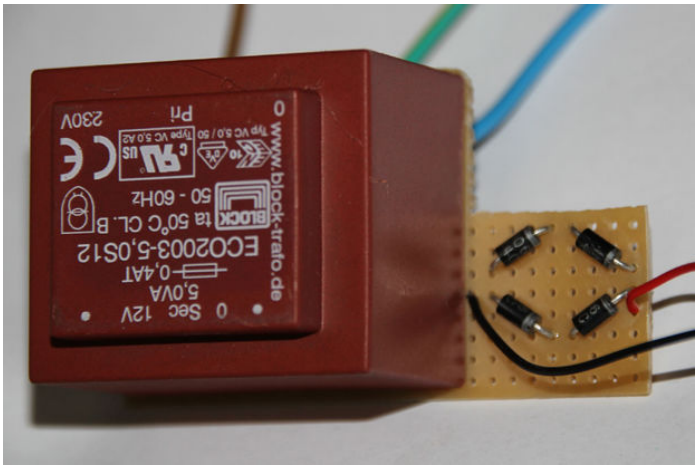
Safety precautions:
In this instructable you'll have to wire some parts with 110 volts line power and most of the heating plate is conductive metal. Unless you are not 100% sure, what you are doing (and even then I would not work without one) it is very advisable to use a ground fault circuit interrupter adapter or a safety socket when working on and with the device. These sockets will shut off the current immediately when a current leakage is present (e.g. through you). Otherwise you have to wait until one of the fuses blows which needs much more power (in Germany line power fuses blow at about 3.6 kW). GFCIs are cheap safety features that I would everybody urge to use. You can buy them e.g. on amazon.com .

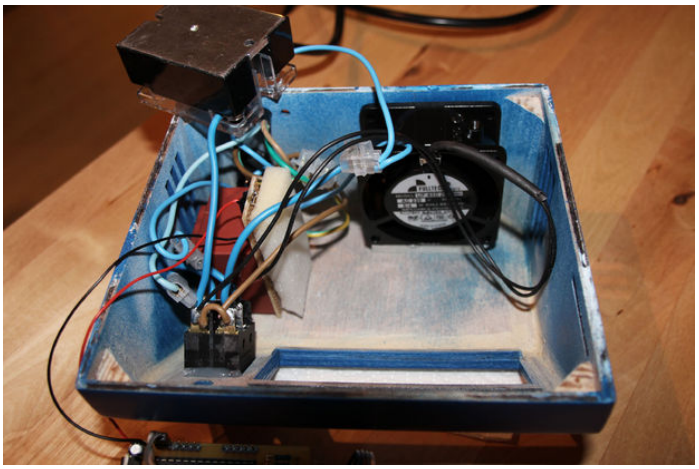
Since I am planning on using heating power up to 1 kW, the main power is led through a 10 amp fuse and a mechanical power switch with a LED power indicator. These devices can disconnect the whole controller and heating circuit from the AC power. The ground wire is fed into the heating cartridge connector and should be attached to all conductive metal parts of the casing.
As you can see in the electronics circuit plan, one of the 110 V lines from the switch is led to the heating cartridge connector through a solid state relay and a second mechanical switch, whereas the second line is wired directly to the heating cartridge connector. When pin 6 on the Atmega controller is set to HIGH and the second mechanical switch is turned on, the load circuit is closed and powers the heating cartridges. The second mechanical switch also contains a LED power indicator which visualizes the heating pulses.
The low voltage supply for the controller circuit is generated by a small print transformer that generates 12 V AC which is subsequently fed into a four diode bridge rectifier and a LM7805 linear voltage regulator.
For measuring the temperature, we use a K-type thermocouple and MAX6675 controller board (MOD-TC). The MAX6675 chip has an internal 12-bit AD converter and can be connected via a serial interface. The LCD (SainSmart IIC/I2C/TWI Serial 2004 20x4 LCD Module Shield), is connected with the Atmega's I2C port and the SD-Card interface board is wired with the SPI interface. For navigation and setting the parameters we use a rotary encoder with integrated push button.
The heating plate also contains a thermal switch, which mechanically disconnects the heating cartridges when the temperature is over 190°C.











File Downloads

PCB_etch_bottom.pdf (8 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'PCB_etch_bottom.pdf']

PCB_etch_top.pdf (25 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'PCB_etch_top.pdf']

Step 5: Code

You can find the code in the attached files. I implemented the following external libraries:
PID library
Bounce library
Rotary Encoder
I2C LC Display
MAX6675

NOTE: In "MAX6675.h" you have to replace line 11 (#include "WProgram.h") with
#if defined(ARDUINO) & ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
in order to make it run on Arduino 1.05

Use the "LiquidCrystal" library in the sub folder 2004-1.0 code, ignore "LiquidCrystal_I2C".

When a temperature is set manually or a program is selected from the SD card, the controller starts reading the temperature ever 0.1 seconds and calculates the next heating pulse length with PID library. As we are using a lot of different libraries I used up more than 99.5% of all the free flash memory on the atmega 328 chip. Worse, I had to shorten the text and menu for the LCD display, as I was running out of SRAM. Next time, I would probably be using an atmega 2560 or swap some strings into the EEPROM.

The different programs have to be saved as comma separated text files with incrementing file names (1.txt,…10.txt) in the root folder of the SD-card. The layout of the files should look like this:
1. Line: title (maximum 20 characters)
2. Line: first temperature set point, first heating rate, first duration
3. Line: second temperature set point, second heating rate, second duration
4. Line ...

You can use up to a maximum of 10 lines per file. It has to be terminated with a blank line.

I used an Arduino UNO to upload the following code to the Atmega:

```
//Temperature Controller
#include <MAX6675.h>

//LCD
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <SD.h>
#include <PID_v1.h>

//Rotary Encoder
#include <Encoder.h>
Encoder roundEnc(2, 3);

//Pushbutton on rotary encoder
#include <Bounce.h>
Bounce pushButton = Bounce(4 ,5);

#define I2C_ADDR    0x3F  // Define I2C Address where the LCD is
#define BACKLIGHT_PIN     3
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

//Temperatursensor
int tCS = 7;               // CS pin on MAX6675
int tSO = 9;               // SO pin of MAX6675
int tSCK = 8;              // SCK pin of MAX6675
int units = 1;              // Units to readout temp (0 = raw, 1 = ËšC, 2 = ËšF)
int TCError = 0;            // ErrorFlag - TC Error
int tempVal = 0;            // Actual-Temperatur

//Display
unsigned long lastMeas = 0;           // Last temperature measurement
unsigned long lastDisp = 0;           // Last display change
unsigned long lastTempDisp = 0;       // Last temperature update on display
unsigned long lastLog = 0;            // Last Logstate
unsigned long logTime = 0;
int lastDispTemp = -1;
int dispOn = 0;                       // blink
int cPosX = -1;                       // x-position (cursor) on display
int cPosY = -1;                       // y-position on display
String emptyString = "                    ";

unsigned long lastDispPrg = 0;        //Zeit seit letztem Update des Programm-Displays

// Has the button on the rotary encoder been pressed?
// 0 = no
// 1 = yes
// 2 = yes, for more than 2 seconds
int buttonPressed = 0;

// Rotational Encoder
```

```
// 0 = not changed
// 1 = clockwise
// 2 = counterclockwise
int rotationState = 0;
int rotationNumber = 0;
int lastRotationNumber = 0;
int rotationBigLeap = 0;     //was the rotationl encoder moved more than ~20°
int oldRotState = 0;

//Program state
// 0: No program
// 1: Manual Setpoint
// 2: Load from SD
// 3: not used
// 4: Program is running
int progState = 0;
int subProgState = 0;
int lastProgState = -1;
int lastSubProgState = -1;

// Programs from SD-Card
int progCount = 0;
byte progLnCount = 0;
byte progTSp[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
byte progSprr[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
word progTime[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

//Which type of program is running at the moment?
//-1 = 0;
//0 = Single-Point
//>1 = SD-Card
int progRun = -1;
int runLine = -1;                     // Actual subprogram of the SDCard
unsigned long runTime = 0;            // How long is the step running already
unsigned long stepTime = 0;           // How much time to increase the ramp
unsigned long lastPuls = 0;           // Last heating puls
int minutesPast = 0;

boolean SDOpen = false;

int aTsp = 0;
int Tsp = 120;              //Setpoint °C
int MaxT = 190;            // Maximal T °C
byte Sprr = 15;               //Rate °C/m
byte MaxR = 20;             // Maximale Ramp
int TAlert = 200;          // Shutown Temperature
int Tdig1 = Tsp/100;                        //Digits for Tsp
int Tdig2 = (Tsp%100)/10;
int Tdig3 = Tsp%10;
int Rdig1 = Sprr/10;                        //Digits for Rp
int Rdig2 = Sprr%10;

// Initialize the MAX6675 Library for our chip
// Setup Serial output and LED Pin
// MAX6675 Library already sets pin modes for MAX6675 chip!
MAX6675 temp(tCS,tSO,tSCK,units);

//PID Parameters
double Setpoint, Input, Output;
PID myPID(&nput, &utput, &etpoint,2,5,1, DIRECT);

void setup() {
 //Serial.begin(9600);

 lcd.begin (20,4);
 // Switch on the backlight
 lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
 lcd.setBacklight(HIGH);
 lcd.noBlink();
 lcd.clear ();

 pinMode(10, OUTPUT);          //SC-Pin SD-Card
 //pinMode(53, OUTPUT);          //SC-Pin SD-Karte - Arduino mega
 //digitalWrite(53, HIGH);

 pinMode(6, OUTPUT);          //heating pin
}

void loop() {
 //disable Heating
 digitalWrite(6, LOW);

 //get user input
 getButtonState();
 getRotState();

 //change display
 setState();

 //Update display
 displayState();

 //Update Temperature
 //Faster measurement when program is running. In the programming stage the LCD lags
 if (progState == 4){
   getTemp();
   updateProgram();
   Input = tempVal;
   Setpoint = aTsp;
   myPID.Compute();
```

```
   heatingPuls();

   if ((lastLog + 1000) < millis()){
     updateLog();
     lastLog = millis();
   }

 }
 else if ((lastMeas + 10000) < millis()){
   getTemp();
   lastMeas = millis();
 }

 if ((tempVal > TAlert) || ( TCError ==1 )) { stopProgram(); delay(5000); }
}

void getButtonState(){
 unsigned long pushTime = pushButton.duration();
 pushButton.update();

 //How long was the button pressed - abort?
 boolean pushed = pushButton.fallingEdge();
 if ((pushed == true) & (pushTime < 1500) ){
   buttonPressed = 1;
 }
 else if ((pushed == true) & (pushTime >= 1500) ){
   buttonPressed = 2;
 }
 else {
   buttonPressed = 0;
 }
}


void getRotState(){
 //int roataionState = 0;
 //int rotationNumber = 0; - Scale 0-10

 long tempState = roundEnc.read();

 if (tempState > oldRotState)
 {
   rotationState = 1;
   //    rotationNumber = (tempState%100)/10;
   int tempRotationNumber = (tempState%100)/10;
   if (tempRotationNumber != rotationNumber) {
     rotationBigLeap = 1;
   }
   rotationNumber = tempRotationNumber;
   if (rotationNumber < 0) {
     rotationNumber = 10 + rotationNumber;
   }

   oldRotState = (tempState%100);
   roundEnc.write(oldRotState);
 }
 else if (tempState < oldRotState)
 {
   rotationState = 2;
   //    rotationNumber = (tempState%100)/10;
   int tempRotationNumber = (tempState%100)/10;
   if (tempRotationNumber != rotationNumber) {
     rotationBigLeap = 1;
   }
   rotationNumber = tempRotationNumber;
   if (rotationNumber < 0) {
     rotationNumber = 10 + rotationNumber;
   }

   oldRotState = (tempState%100);
   roundEnc.write(oldRotState);
 }
 else  {
   rotationState = 0;
   rotationBigLeap = 0;
 }
}

void resetRotState(int digit){
 //For entering numbers in the LCD
 rotationNumber = digit;
 //Failsave
 if ((rotationNumber > 10) || (rotationNumber < 0)) {
   rotationNumber = 0;
 }
 oldRotState = digit * 10;
 roundEnc.write(oldRotState);
}

//Verarbeite Eingaben am Display
void setState(){
 //Rotationstate 1 = right
 //Rotationstate 2 = left

 //Long pressing: abort
 if ((progState != 0) && (buttonPressed == 2)) {
   stopProgram();
   buttonPressed = 0;
   return;
 }
```

```
//submenu 0,1
if ((progState == 0) && (rotationState != 0)){
  int select = rotationNumber%2 + 1;
  subProgState = select;
  displayState();
  return;
}

//choose menu entry
//manual setpoint
if ((progState == 0) && (buttonPressed == 1)){
  if (subProgState == 1) {
    progState = 1;
    subProgState = 0;
    displayState();
    buttonPressed = 0;
    return;
  }
 //SD-Card
  if (subProgState == 2) {
    progState = 2;
    subProgState = 0;
    displayState();
    buttonPressed = 0;
    return;
  }
}

//Submenue 0,1 - manual mode
if ((progState == 1) & (subProgState < 4) && (rotationState == 1) && (rotationBigLeap == 1)){
  subProgState = subProgState + 1;
  if (subProgState > 3) {
    subProgState = 0;
  }
  displayState();
  rotationBigLeap = 0;
  return;
}
if ((progState == 1) & (subProgState < 4) && (rotationState == 2) && (rotationBigLeap == 1)){
  subProgState = subProgState - 1;
  if (subProgState < 0) {
    subProgState = 3;
  }
  displayState();
  rotationBigLeap = 0;
  return;
}

//Back, TSP, Rp, Start in Manual Mode
if ((progState == 1) && (subProgState == 0) && (buttonPressed == 1)) {
  stopProgram();
  buttonPressed = 0;
  return;
}
if ((progState == 1) && (subProgState == 1) && (buttonPressed == 1)) {
  subProgState = 11;
  buttonPressed = 0;
  return;
}
if ((progState == 1) && (subProgState == 2) && (buttonPressed == 1)) {
  subProgState = 21;
  buttonPressed = 0;
  return;
}
if ((progState == 1) && (subProgState == 3) && (buttonPressed == 1)) {
  startProgram(0);
  buttonPressed = 0;
  return;
}

//1., 2., 3. Digit T-Setpoint
if ((progState == 1) && (subProgState == 11) && (buttonPressed == 1)) {
  subProgState = 12;
  buttonPressed = 0;
  return;
}
if ((progState == 1) && (subProgState == 12) && (buttonPressed == 1)) {
  subProgState = 13;
  buttonPressed = 0;
  return;
}
if ((progState == 1) && (subProgState == 13) && (buttonPressed == 1)) {
  //Last digit
  subProgState = 1;
  Tsp = Tdig1 * 100 + Tdig2 * 10 + Tdig3;
  if (Tsp > MaxT) {
    Tsp = MaxT;
  }      //Maximal Temperature
  Tdig1 = Tsp/100;                        //Digits for Tsp
  Tdig2 = (Tsp%100)/10;
  Tdig3 = Tsp%10;

  buttonPressed = 0;
  return;
}

//1., 2. Digit ramp
```

```
 if ((progState == 1) && (subProgState == 21) && (buttonPressed == 1)) {
   subProgState = 22;
   buttonPressed = 0;
   return;
 }
 if ((progState == 1) && (subProgState == 22) && (buttonPressed == 1)) {
   //Last digit
   subProgState = 2;
   Sprr = Rdig1 * 10 + Rdig2;
   if (Sprr > MaxR) {
     Sprr = MaxR;
   }      //Maximal/minimal ramp
   if (Sprr < 1) {
     Sprr = 1;
   }
   Rdig1 = Sprr/10;                        //Digits for Rp
   Rdig2 = Sprr%10;

   buttonPressed = 0;
   return;
 }

 //Submenu 1,1 - SD-Programm
 if ((progState == 2) & (subProgState < 11) && (rotationState == 1) && (rotationBigLeap == 1)){
   subProgState = subProgState + 1;
   if (subProgState > progCount) {
     subProgState = 1;
   }
   displayState();
   rotationBigLeap = 0;
   return;
 }
 if ((progState == 2) & (subProgState < 11) && (rotationState == 2) && (rotationBigLeap == 1)){
   subProgState = subProgState - 1;
   if (subProgState < 1) {
     subProgState = progCount;
   }
   displayState();
   rotationBigLeap = 0;
   return;
 }

 if ((progState == 2) & (subProgState < 11) && (buttonPressed == 1)){
   startProgram(subProgState);
   buttonPressed = 0;
   return;
 }
}

void stopProgram(){
 progState = 0;
 subProgState = 0;
 progRun = -1;
 runLine = -1;
 lastSubProgState = -1;
 lastProgState = -1;
 displayState();
 myPID.SetMode(MANUAL);
}

void startProgram(int runType){
 progRun = runType;
 aTsp = tempVal;
 runTime = millis();
 runLine = 0;
 progState = 4;
 myPID.SetMode(AUTOMATIC);
 logTime = millis();
}

void updateProgram()
{
 //SD-Card
 //progLnCount
 //runline
 minutesPast = 1.0 * ((millis() - runTime) / 60000.0);
 if (progRun > 0){
   Tsp = progTSp[runLine];
   Sprr = progSprr[runLine];

   //Load new program?
   if (minutesPast >= progTime[runLine])
   {
     runLine++;
     if (runLine > progLnCount) { stopProgram(); return; }
     runTime = millis();
   }
 }

 //Ramp
 long timeStep = (1000.0 * 60.0) / (double)Sprr;

 if ((stepTime + timeStep < millis()) & (aTsp < Tsp)) { aTsp++; stepTime=millis(); }
 else if ((stepTime + timeStep < millis()) & (aTsp > Tsp)) { aTsp--; stepTime=millis(); }
}

void heatingPuls(){
 //pulse every 0.1 s
 if ((lastPuls + 100) < millis()) {
```

```
   digitalWrite(6, HIGH);
   int dly = Output * 1;
   //limit Power
   if (dly > 1000) { dly = 1000; }
   else if (dly < 0) { dly = 0; }
   //Serial.println(dly);
   delay(dly);
   digitalWrite(6, LOW);
   lastPuls = millis();
 }
}

void getTemp(){
 // Read the temp from the MAX6675
 float loop_temperature[2];
 for (int ii=0; ii<2; ii++)
 {
   loop_temperature[ii] = temp.read_temp();
   if (loop_temperature[ii] < 0) {
     TCError = 1;
   }
   delay(50);
 }
 tempVal = (int)((loop_temperature[0] + loop_temperature[1] ) / 2 + 0.5);
}


void updateTemp( boolean force = false, int lastPosX = -1, int lastPosY = -1 ){
 if (((millis() > (lastTempDisp + 1000)) && ( tempVal != lastDispTemp)) || (force == true))
 {
   lcd.setCursor(0, 3);
   lcd.print("T: ");
   lcd.setCursor(3, 3);
   lcd.print(tempVal);
   lcd.print((char)223);
   lcd.print("C   ");
   lastTempDisp = millis();
   lastDispTemp = tempVal;

   if ((lastPosX != -1) || (lastPosY != -1)) {
     lcd.setCursor(lastPosX, lastPosY);
   }
 }
}

void fillNumber3(int number, String &dest3){
 if (number < -9) { dest3 += String(number); }
 else if (number < 0) { dest3 = "0"; dest3 += String(number); }
 else if (number < 10) { dest3 = "00"; dest3 += String(number); }
 else if (number < 100) { dest3 = "0"; dest3 += String(number); }
 else {
   dest3 = String(number);
 }
}

void fillNumber2(int number, String &dest2){
 if (number < 10) {
   dest2 = "0";
   dest2 += String(number);
 }
 else {
   dest2 = String(number);
 }
}

void printProgLine(byte number, String &line){
 line = (String)(number);
 line += ": ";
 line += (String)(progTSp[number]);
 line += ", ";
 line += (String)(progSprr[number]);
 line += ", ";
 line += (String)(progTime[number]);
 line += "    ";
}

void printProcessLine(String &line){
 String saSP;
 fillNumber3( aTsp, saSP);
   String st;
 fillNumber3( minutesPast , st);
   line = "TaSp: ";
 line += saSP;
 line += (char)223;
 line += "C ";
 line += " t: "; line += st;
}

void printSingleLine(String &line){
 String sTsp;
 String sSprr;
 fillNumber3(Tsp, sTsp);
 fillNumber2(Sprr, sSprr);

 line = "Tsp: ";
 line += sTsp;
 line += (char)223;
 line += "C ";
 line += "Rp: ";
```

```
 line += sSprr;
 line += (char)223;
 line += "/m";
}

//Update display
void displayState(){
 if (progState==0) {
   display0();
 }
 if (progState==1) {
   display1();
 }
 if (progState==2) {
   display2();
 }
 if (progState==4) {
   display4();
 }
}

void display0(){
 if (lastProgState != 0){
   cPosX = -1;
   cPosY = -1;
   lcd.noBlink();
   lcd.clear ();                         // go home
   lcd.print("Program");      // program ended
   lastProgState = 0;
   lastSubProgState = -1;          //Reset
   updateTemp(true);              //Update Temperature
 }

 //Update every second
 if ((subProgState == 0) & (millis() > (lastDisp + 1000)) && (dispOn == 0))
 {
   lcd.setCursor(0, 1);        // go to the 2nd line
   lcd.print("-");
   lastDisp = millis();
   dispOn = 1;
 }
 else if ((subProgState == 0) & (millis() > (lastDisp + 1000))){
   lcd.setCursor ( 0, 1 );        // go to the 2nd line
   lcd.print(emptyString);
   lastDisp = millis();
   dispOn = 0;
 }
 else if ((subProgState == 1) && (lastSubProgState != 1)){
   lcd.setCursor(0, 1);
   lcd.print("Setpoint");
   lastSubProgState = 1;
   updateTemp(true);
 }
 else if ((subProgState == 2) && (lastSubProgState != 2)){
   lcd.setCursor(0, 1);
   lcd.print("SDCard  ");
   lastSubProgState = 2;
   updateTemp(true);
 }

 updateTemp();
}

//Manual setpoint
void display1(){
 if (lastProgState != 1){
   cPosX = -1;
   cPosY = -1;
   lcd.noBlink();
   lcd.clear ();                          // go home
   lcd.print("Setpoint");
   lcd.setCursor(0, 1);
   String line;
   printSingleLine(line);
   lcd.print(line);

   lastProgState = 1;
   lastSubProgState = -1;
   updateTemp(true);
 }

 //0. Back
 if ((subProgState == 0) && (lastSubProgState != 0)) {
   lcd.noBlink();
   lcd.setCursor(0, 2);
   lcd.print("Back ");
 }

 //1. Set Temp
 if ((subProgState == 1) && (lastSubProgState != 1)) {
   lcd.noBlink();
   lcd.setCursor(0, 1);
   String line;
   printSingleLine(line);
   lcd.print(line);
   lcd.setCursor(0, 2);
   lcd.print("TSp  ");
   lastSubProgState = 1;
 }
```

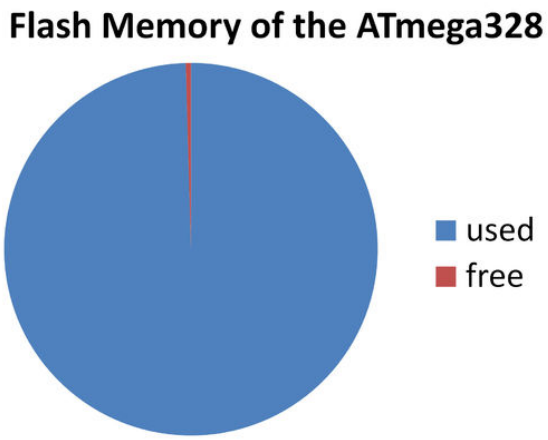
```
//2. Set Ramp
if ((subProgState == 2) && (lastSubProgState != 2)) {
  lcd.noBlink();
  lcd.setCursor(0, 1);
  String line;
  printSingleLine(line);
  lcd.print(line);

  lcd.setCursor(0, 2);
  lcd.print("Rp   ");
  lastSubProgState = 2;
}

//3. Start
if ((subProgState == 3) && (lastSubProgState != 3)) {
  lcd.noBlink();
  lcd.setCursor(0, 2);
  lcd.print("Start");
  lastSubProgState = 3;
}

//4. Change T 1. Digit
if ((subProgState == 11) && (lastSubProgState != 11)) {
  //1. Digit cPosX = 1, cPosY = 1
  cPosX = 5;
  cPosY = 1;
  lcd.blink();
  resetRotState(Tdig1);
  lcd.setCursor(cPosX, cPosY);
  lcd.print(Tdig1);
  lcd.setCursor(cPosX, cPosY);
  lastSubProgState = 11;
}
else if (subProgState == 11) {
  lcd.setCursor(cPosX, cPosY);
  Tdig1 = rotationNumber;
  lcd.print(Tdig1);
  lcd.setCursor(cPosX, cPosY);
}

//5. Change T 2. Digit
if ((subProgState == 12) && (lastSubProgState != 12)) {
  //2. Digit cPosX = 6, cPosY = 1
  cPosX = 6;
  cPosY = 1;
  lcd.blink();
  resetRotState(Tdig2);
  lcd.setCursor(cPosX, cPosY);
  lcd.print(Tdig2);
  lcd.setCursor(cPosX, cPosY);
  lastSubProgState = 12;
}
else if (subProgState == 12) {
  lcd.setCursor(cPosX, cPosY);
  Tdig2 = rotationNumber;
  lcd.print(Tdig2);
  lcd.setCursor(cPosX, cPosY);
}

//6. Change T 3. Digit
if ((subProgState == 13) && (lastSubProgState != 13)) {
  //3. Digit cPosX = 7, cPosY = 1
  cPosX = 7;
  cPosY = 1;
  lcd.blink();
  resetRotState(Tdig3);
  lcd.setCursor(cPosX, cPosY);
  lcd.print(Tdig3);
  lcd.setCursor(cPosX, cPosY);
  lastSubProgState = 13;
}
else if (subProgState == 13) {
  lcd.setCursor(cPosX, cPosY);
  Tdig3 = rotationNumber;
  lcd.print(Tdig3);
  lcd.setCursor(cPosX, cPosY);
}

//4. Change Ramp 1. Digit
if ((subProgState == 21) && (lastSubProgState != 21)) {
  //1. Digit cPosX = 15, cPosY = 1
  cPosX = 15;
  cPosY = 1;
  lcd.blink();
  resetRotState(Rdig1);
  lcd.setCursor(cPosX, cPosY);
  lcd.print(Rdig1);
  lcd.setCursor(cPosX, cPosY);
  lastSubProgState = 21;
}
else if (subProgState == 21) {
  lcd.setCursor(cPosX, cPosY);
  Rdig1 = rotationNumber;
  lcd.print(Rdig1);
  lcd.setCursor(cPosX, cPosY);
}
```

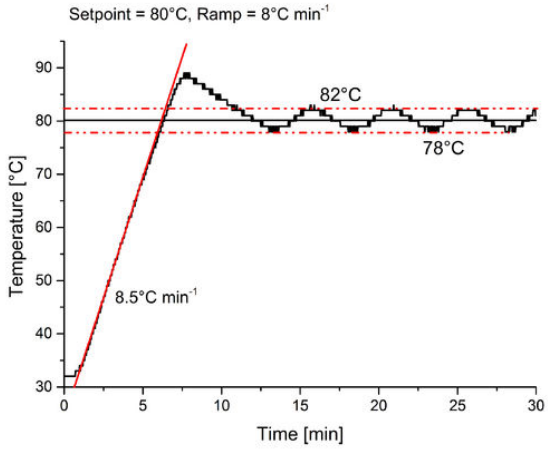
```
 //5. Change Ramp 2. Digit
 if ((subProgState == 22) && (lastSubProgState != 22)) {
   //2. Digit cPosX = 16, cPosY = 1
   cPosX = 16;
   cPosY = 1;
   lcd.blink();
   resetRotState(Rdig2);
   lcd.setCursor(cPosX, cPosY);
   lcd.print(Rdig2);
   lcd.setCursor(cPosX, cPosY);
   lastSubProgState = 22;
 }
 else if (subProgState == 22) {
   lcd.setCursor(cPosX, cPosY);
   Rdig2 = rotationNumber;
   lcd.print(Rdig2);
   lcd.setCursor(cPosX, cPosY);
 }

 updateTemp(false, cPosX, cPosY);
}

void display2(){
 if (lastProgState != 2){
   lcd.noBlink();
   lcd.clear ();                          // go home
   lcd.print("SDCard");
   lcd.setCursor(0, 1);

   if (SDOpen != true) {
     if (! SD.begin(10)) { stopProgram(); return; }
   }
   SDOpen = true;
   progCount = 0;
   for (int ii = 1; ii < 10; ii++){
     String fname = String(ii);
     fname += ".txt";
     char fnamec[6];
     fname.toCharArray(fnamec, 6);

     File myFile = SD.open(fnamec, FILE_READ);

     if (myFile) {
       progCount++;
     }
     myFile.close();
   }

   if (progCount == 0) {
     stopProgram();
     return;
   }

   lastProgState = 2;
   subProgState = 1;
   lastSubProgState = -1;
   updateTemp(true);
 }

 if ((subProgState < 10) && (lastSubProgState != subProgState))
 {
   String fname = String(subProgState);
   fname += ".txt";
   char fnamec[6];
   fname.toCharArray(fnamec, 6);

   File myFile = SD.open(fnamec, FILE_READ);
   if (! myFile) {
     stopProgram();
   }

   byte line = 0;
   byte Slength = 0;
   char tempString[1];
   String HeadLine = "";
   String ProgLine = "";

   for (byte ii=0; ii < 10; ii++)
   {
     progTSp[ii] = 0;
     progSprr[ii] = 0;
     progTime[ii] = 0;
   }

   while (myFile.available()) {
     tempString[1] = myFile.read();
     if (tempString[1] == '\r' ) {
     }
     else if (tempString[1] != '\n' ) {
       ProgLine += tempString[1];
       Slength++;
     }
     else {
       if (line == 0) { HeadLine = ProgLine; ProgLine = ""; Slength = 0; line++; }
       else {
         if ((line > 10) || (Slength > 20) ) {
           stopProgram();
           return;
         }
```

```
        char* accum;
        char buffer[Slength+1];
        ProgLine.toCharArray(buffer, Slength+1);
        const char sep[] = ",";

        progTSp[line-1] = atoi(strtok_r(buffer, sep, &ccum));
        if (progTSp[line-1] > MaxT) {
          progTSp[line-1] = MaxT;
        }
        progSprr[line-1] = atoi(strtok_r(NULL, sep, &ccum));
        if (progSprr[line-1] > MaxR) {
          progSprr[line-1] = MaxR;
        };
        if (progSprr[line-1] < 1) {
          progSprr[line-1] = 1;
        }
        progTime[line-1] = atoi(strtok_r(NULL, sep, &ccum));
        progLnCount = line - 1; //0-Anzahl-1

        Slength = 0;
        line++;
        ProgLine = "";
        //
      }
    }
  }
  myFile.close();

  lcd.setCursor(0, 1);
  lcd.print(emptyString);
  lcd.setCursor(0, 1);
  lcd.print(HeadLine);
  printProgLine(0, ProgLine);
  lcd.setCursor(0, 2);
  lcd.print(emptyString);
  lcd.setCursor(0, 2);
  lcd.print(ProgLine);

  resetRotState(0);
  lastSubProgState = subProgState;
 }
 updateTemp();
}


void display4(){
 if ((lastDispPrg + 1000) > millis()) { return; }

 if (lastProgState != 4) {
 lcd.noBlink();
 lcd.clear ();
 updateTemp(true);
 lastProgState = 4;
 }

 if ( progRun == 0 )
 {
   lcd.setCursor(0, 0);
   String line;  printProcessLine(line);
   lcd.print(line);

   lcd.setCursor(0, 1);
   printSingleLine(line);
   lcd.print(line);
 }

 if ( progRun > 0 )
 {
   lcd.setCursor(0, 0);
   String line;  printProcessLine(line);
   lcd.print(line);

   lcd.setCursor(0, 1);
   printProgLine(runLine, line);

   lcd.print(line);
   if (runLine != progLnCount) {
     lcd.setCursor(0, 2);
     printProgLine(runLine+1, line);
     lcd.print(line);
   }
   else {
     lcd.setCursor(0, 2);
     lcd.print(emptyString);
   }
 }

 updateTemp();
}

void updateLog(){
  if (SDOpen != true) {
     if (! SD.begin(10)) { return; }
   }
   SDOpen = true;

   File myFile = SD.open("T.log", FILE_WRITE);

   if (! myFile) { return; }
```

```
   myFile.print(millis() - logTime);
//    myFile.print(", ");
//    myFile.print(aTsp);
   myFile.print(", ");
   myFile.println(tempVal);
   myFile.close();
}
```

As you can see in the picture, for a first (and not optimized) try these are very nice results. A really nifty way to improve the system even more would be to implement an autotunig library (e.g. from here), but it would need more space than what is available on the atmega328. An other optimization would be to use a more accurate thermocouple controller.

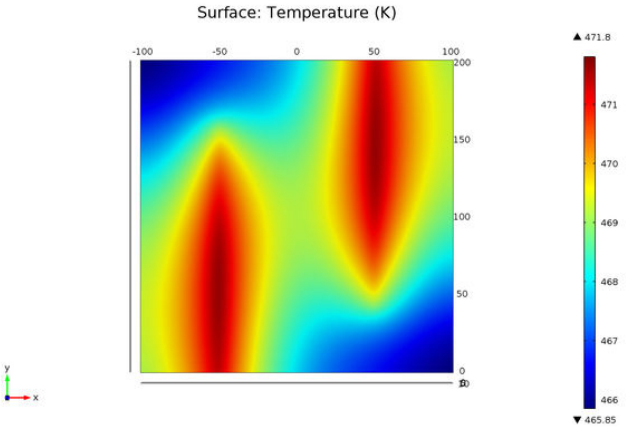


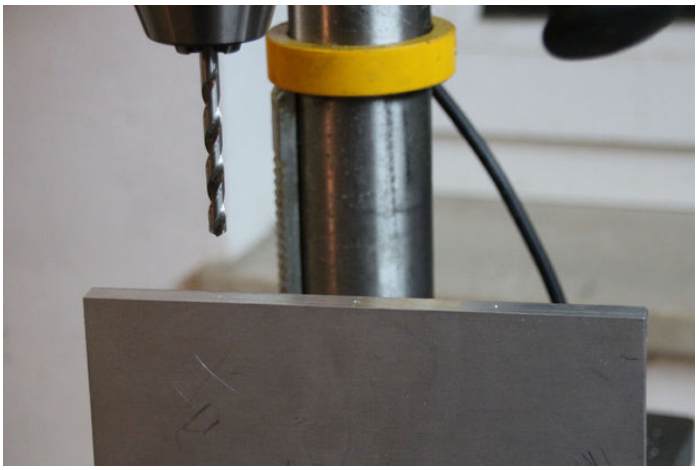


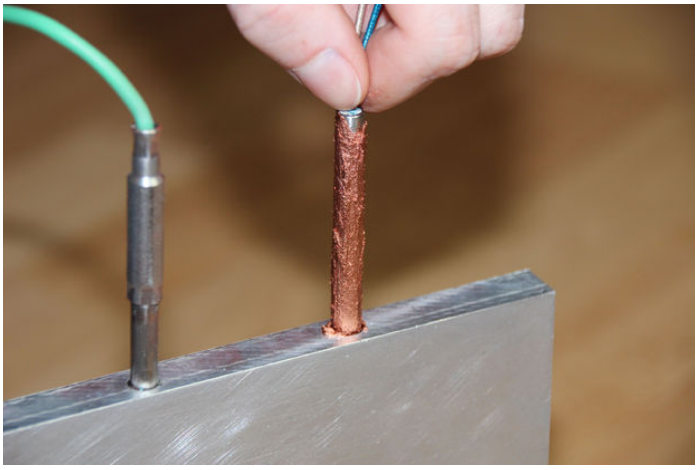
Step 6: Preparing the hot plate

To find out, which the best position for the heating cartridges is and whether an adequate heat distribution can be reached using 1 cm thick aluminum, I simulated the stationary temperature profile. As you can see, at about 470 K (197°C, 387 F) the local deviation is roughly ±3 K which is enough for my purpose.

Start by drilling the holes for the heating cartridges and the thermocouple. This part is a little bit tricky and you definitely need a drill press for this step. You also can try to use two aluminum plates and routing machine with a ball nose cutter. Next, drill the holes for the steel thread rods and locking screws into the aluminum plates and metal support. If you want to use a thermal switch as safety device, also drill the matching holes into the aluminum plate. Lubricate the heating cartridge and thermocouple with the copper paste to improve the heat transfer to the aluminum. Afterwards use lock screws to hold them in the plate.





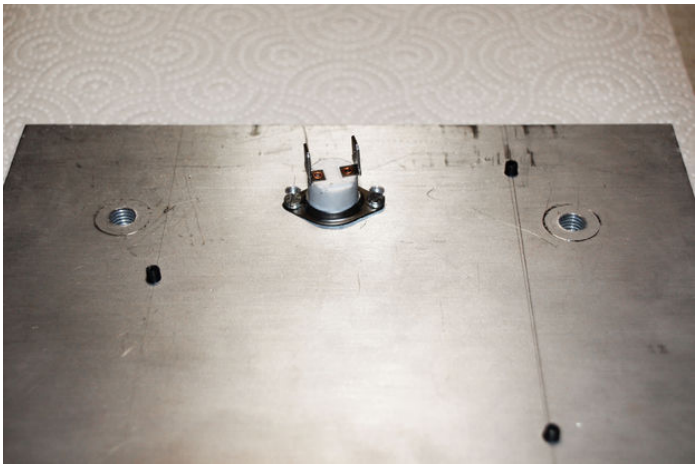


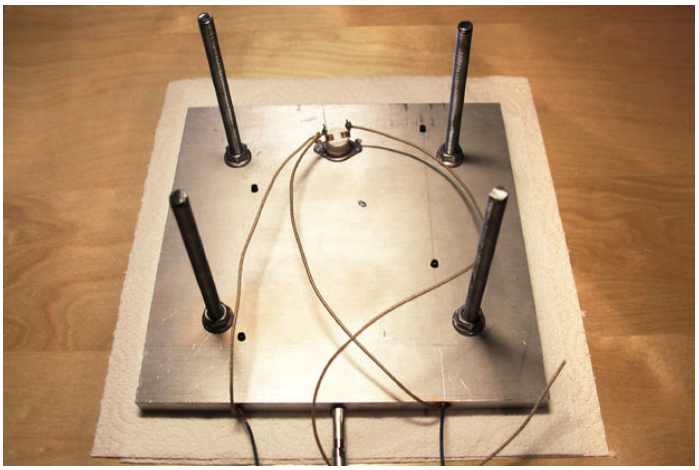
Step 7: Assembling the hot plate

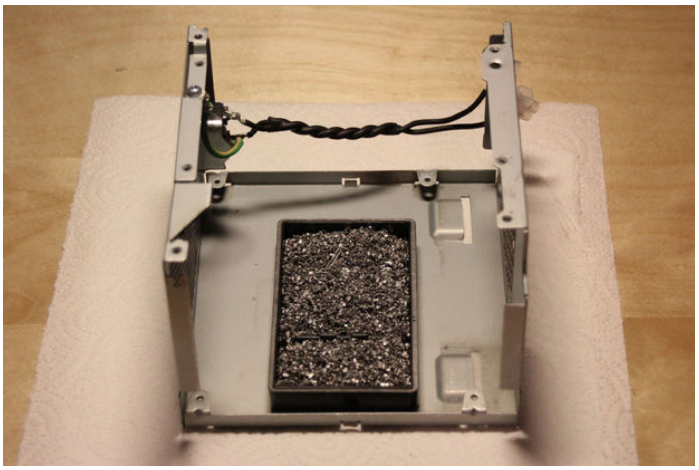
Fix the thermal switch to the plate and solder one wire of each cartridge to one pin of the switch. Connect the other pin to the plug in the metal casing and the other wire directly to a pin of the metal case.
To stabilize the heating plate, you can glue a small plastic casing into the metal support plate and fill it with lead.

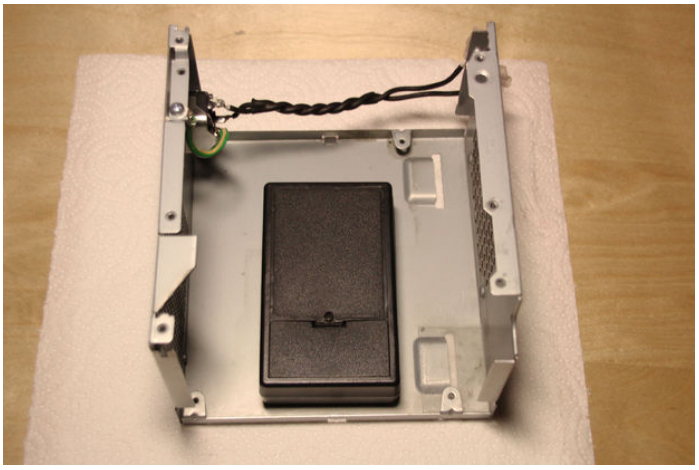
In order to get an impression how hot the metal casing of the support gets, I simulated the temperature profile of the heating plate and the steel rods below the plate. It turns out that at 540 K (270°C, 518 F) the steel rods are about 370 K (97 °C, 206 F) hot. Since I limit the temperature to 180°C, I could also have used a wooden casing.

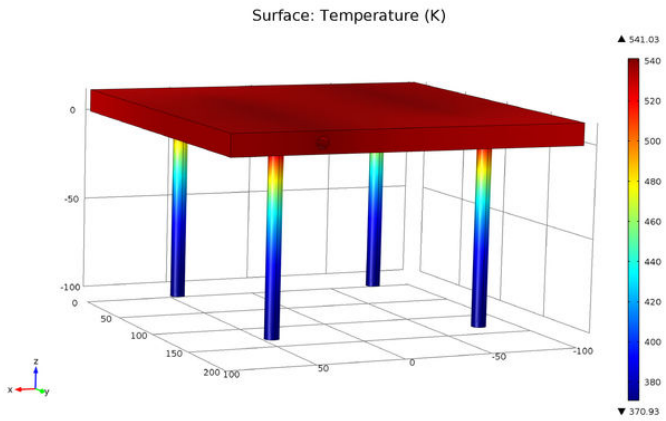
I placed a warning on the hot plate with heat resistant spray paint and used the hot plate to burn it onto the aluminum.











Related Instructables

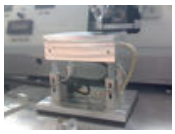
A TEMPERATURE CONTROLLED LABORATORY HEATER by kymyst

Magnetic Stirrer Hotplate by kymyst

"MELT-O-MATIC" DIGITAL MELTING POINT APPARATUS by kymyst

Slide Ringing Table by kymyst

Making A Thermocouple by trebuchet03



Heated Stage for Thermosonic Wedge Bonding by kc6qhp

Comments

23 comments **Add Comment**

clazman says: Dec 3, 2013. 7:44 PM **REPLY**

At the start of the read I wanted to say PID control, then utilize auto learning , You are using PID and are considering the latter. Kudos to you.

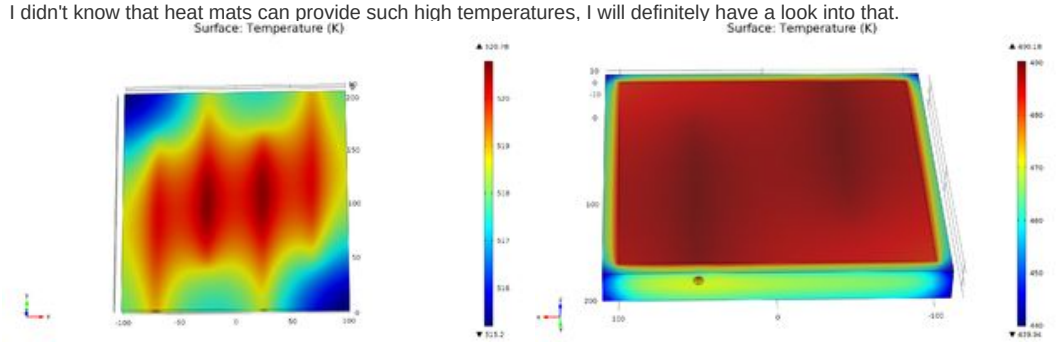
PID is required and auto learning is great. But both features are dependent on the heat load. So several heating profiles will be required.

On the other hand, I am not very happy with the plates temperature profile. More cartridges are needed. I don't know if there are variable wattage cartridge heaters in th\is size range. I've used that feature in say 5/8" X 24" heaters to help create a more uniform temp. profile in rubber molding platens. Have you considered heat mats? They could help in heat distribution.

Again, I am still impressed with your prototype design.

BrittLiv says: Dec 4, 2013. 1:48 PM **REPLY**

Thank you! 4 cartridge heaters are not going to change as much as you might hope. I have uploaded another simulation and since you could be interested one after applying insulation.
In my case two are really enough, since I doubt, that I will ever heat something of exactly the same size as the the plate.
I didn't know that heat mats can provide such high temperatures, I will definitely have a look into that.



hbouwens says: Dec 4, 2013. 1:34 AM **REPLY**

there is another way to save the program steps. I use one line of text as in :

6,300,130,60,780,600,60,1,700,140,1,510,90,600,400,20,900,20,0
With no maximum to the number of steps

The first number is the number of steps, the rest is in blocks of 3, the rate, Temperature and rest-time. With a function I read these values int an array.

I could upload the code and a picture , but apparently I don't know how to :(

chutchinson1 says: Dec 4, 2013. 4:11 AM **REPLY**

I haven't checked your code (sorry) but have you thought about quantizing your temperature so it is only specified in steps of 5 for example. This way you can use a uint8_t (i.e. unsigned char). Doing this with all the data you might want to use a C bit-field (http://en.wikipedia.org/wiki/Bit_field)

You may also could reduce the memory usage further with a basic zip/delta compression. A simple approach you store the delta/change (+-128) between steps so that each value can be stored as a int8_t (i.e. signed char) . This incurs a slight increase in complexity and requires extra steps for large step changes. There are many ways you could take this further including RLE encoding and a 3bits step header which determines what has changed between steps. Complexity of the code goes up a significantly there though so would probably never be worth the effort!

hbouwens says: Dec 4, 2013. 5:12 AM **REPLY**

You couldn't check as I didn't upload it so far. If you could tell me how to upload files, You could see what the code is doing, and...how "good" the wanted curve is followed :)

BrittLiv says: Dec 4, 2013. 2:56 AM **REPLY**

Thanks, but the step limit is actually given by the array-space I allocate in the Atmega. When running a program I want to load the whole program into the RAM so you can remove the SD card without canceling your run

hbouwens says: Dec 4, 2013. 5:03 AM **REPLY**

Dear BrittLiv, how can I upload a pdf and ino file as comment? I'm new to Instructables, and don't know how to
Pushing the button add Images and select files seems not to be enough :(

BrittLiv says: Dec 4, 2013. 5:51 AM **REPLY**

Hi, thanks a lot for your effort! As far as I know you can't add ino files to instructables.
That's why I used a text box. Since you are a pro member, just click on "Rich Editor" then "Source" and add one like this:
<pre><textarea cols="75" name="code" readonly="" rows="20" style="width: 600.0px;height: 294.0px;">**your code** </textarea></pre>

About the pdf I am not sure whether you can add it to comments. Can you maybe upload it somewhere else (e.g. dropbox) and share a link?

The simplest solution to the space problem is probably to get a second Atmega connect it with the IC2 interface and to split all the tasks. Especially since the SC-Card library literally eats up your space (~16 kByte) and the spare amount of free SRAM (apparently only a few bytes are left, as initializing new variables kills the program) might interfere with most of the more sophisticated approaches.

hbouwens says: Dec 4, 2013. 8:42 AM **REPLY**

I don't know why but now the selecting of files worked

You will find my .ino file, a pdf with the result (in Dutch but the graph is International :)), a photo of the oven and control and last but not least a curve file.

By the way it is a setup to heat glass , to shape and fuse

regards, Harrie

StookCurve.pdf105 KB

OvenControll.pdf854 KB

Curv07.txt63 bytes

OvenCurve14jun13.txt9 KB

BrittLiv says: Dec 4, 2013. 11:53 AM **REPLY**
Wow, great job! Thanks for sharing your code! I will definitely take a look at it.

dvdwlmn says: Dec 4, 2013. 12:45 AM **REPLY**
Wow - very impressed with the 328 coding in this project! i will have to read through it carefully again to really lean the most from this one!!

mbaker750 says: Dec 3, 2013. 1:55 PM **REPLY**
Awesome writeup!

The autotunig library you mentioned would be to fine tune the PID parameters and help prevent the slight overshoot of the initial temperature ramp?

What software did you use for the temperature simulations?

BrittLiv says: Dec 3, 2013. 2:38 PM **REPLY**
Hi, thank you! The autotuning library automatically determines tuning constants. You can find out more about it here.
I used COMSOL Multiphysics.

mbaker750 says: Dec 3, 2013. 3:07 PM **REPLY**
Great! Thanks for the reply (and link)!

bwheat says: Dec 3, 2013. 1:51 PM **REPLY**
Your instructables are always amazing and professional!!! Fantastic Work!!!

PS... Voted :)

BrittLiv says: Dec 3, 2013. 2:39 PM **REPLY**
Thank you! Getting feedback like yours motivates me a lot!

wittend says: Dec 3, 2013. 11:19 AM **REPLY**
I could use more description of the 'heating cartridges'. I am not immediately familiar with the type shown. Where would I expect to find these?

BrittLiv says: Dec 3, 2013. 11:39 AM **REPLY**
Hi, you can get them for example on amazon.com or from China on ebay.com. Thanks for bringing that to my attention, I have added a link to the parts list.

lapsmith says: Dec 3, 2013. 11:03 AM **REPLY**
Great instructable. I could use it for my heat treating furnace, if and when I get the time to build it. Voted for you!

BrittLiv says: Dec 3, 2013. 11:25 AM **REPLY**
Thank you! I would love to see some pictures, should you finish it.

padbravo says: Dec 2, 2013. 9:51 AM **REPLY**
good for the "caution advice" (Murphy's law: "a hot surface its exactly like a cold surface")

TP_inc says: Dec 1, 2013. 2:54 PM **REPLY**
I thought the bottom of the hot plate looked like something out of the computer I am taking apart

BrittLiv says:
You are right, I recycled an old computer power supply.

Dec 1, 2013. 2:58 PM **REPLY**