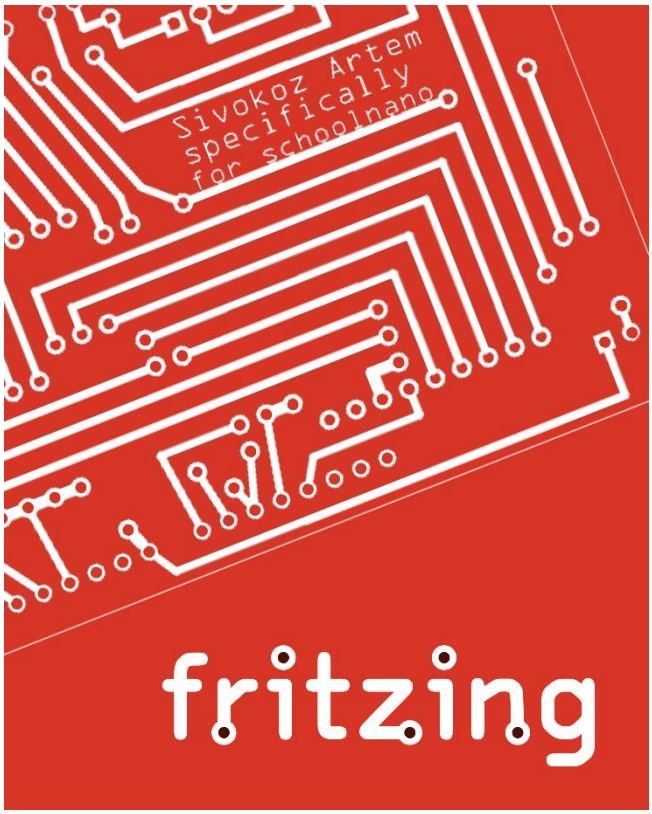


ARDUINO ЙОГУРТНИЦА

Специально для «Школы на ладони»
на конкурс «Удивительный Фритзинг»

ПРОЕКТ ПОДГОТОВИЛ:
Ученик 5 «Л» класса
ТМОЛ г. Таганрога
Сивокоз Артем Владиславович

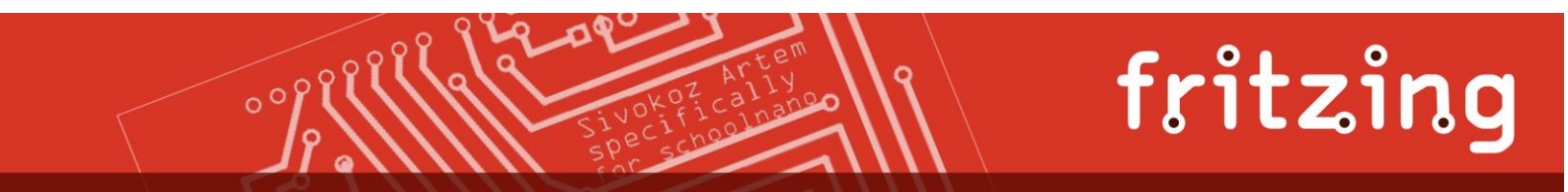
В весенней сессии школы на ладони я решил принять участие в конкурсной программе “Некислая история” и разработать свой собственный уникальный рецепт йогурта, а на основе выбранной технологии устроить домашнюю, биотехнологическую мини-лабораторию по его производству. Для этих целей мне понадобилась собственная йогуртница. Притом не простая, а с возможностью программирования.

Суть йогуртницы очень проста, подготовленная к брожению смесь (в простом варианте это молоко и молочнокислые бактерии, бифидобактерии, обладающие пробиотическими свойствами) выдерживается определенное время при определенной температуре.

Для создания собственной мини-лаборатории, мне потребуется йогуртница с регулировками температуры и длительности брожения. Для управления процессом приготовления было решено использовать программируемый микроконтроллер Arduino. Это не первый мой проект с использованием Arduino, поэтому я уже знаю, на что она способна. У меня в наличии были Arduino UNO, монтажная плата, радиодетали и множество проводов.

Arduino UNO самая популярная плата имеющая 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов. С ее помощью я легко смогу осуществить свои планы.

Самое сложное в этой задаче придумать, как осуществлять непрерывный нагрев баночки с заданной температурой. Если поставить баночку на нагревательный элемент, тогда баночка будет прогреваться не равномерно. Можно поставить баночку внутрь емкости и нагревать эту емкость поддерживая нужную температуру воздуха внутри, но тогда нам понадобится герметичная емкость и нагревательный элемент, а можно поместить баночку в воду и поддерживать заданную температуру воды. Последний вариант оказался самым лучшим, т.к. для этих целей можно использовать простой электрочайник.

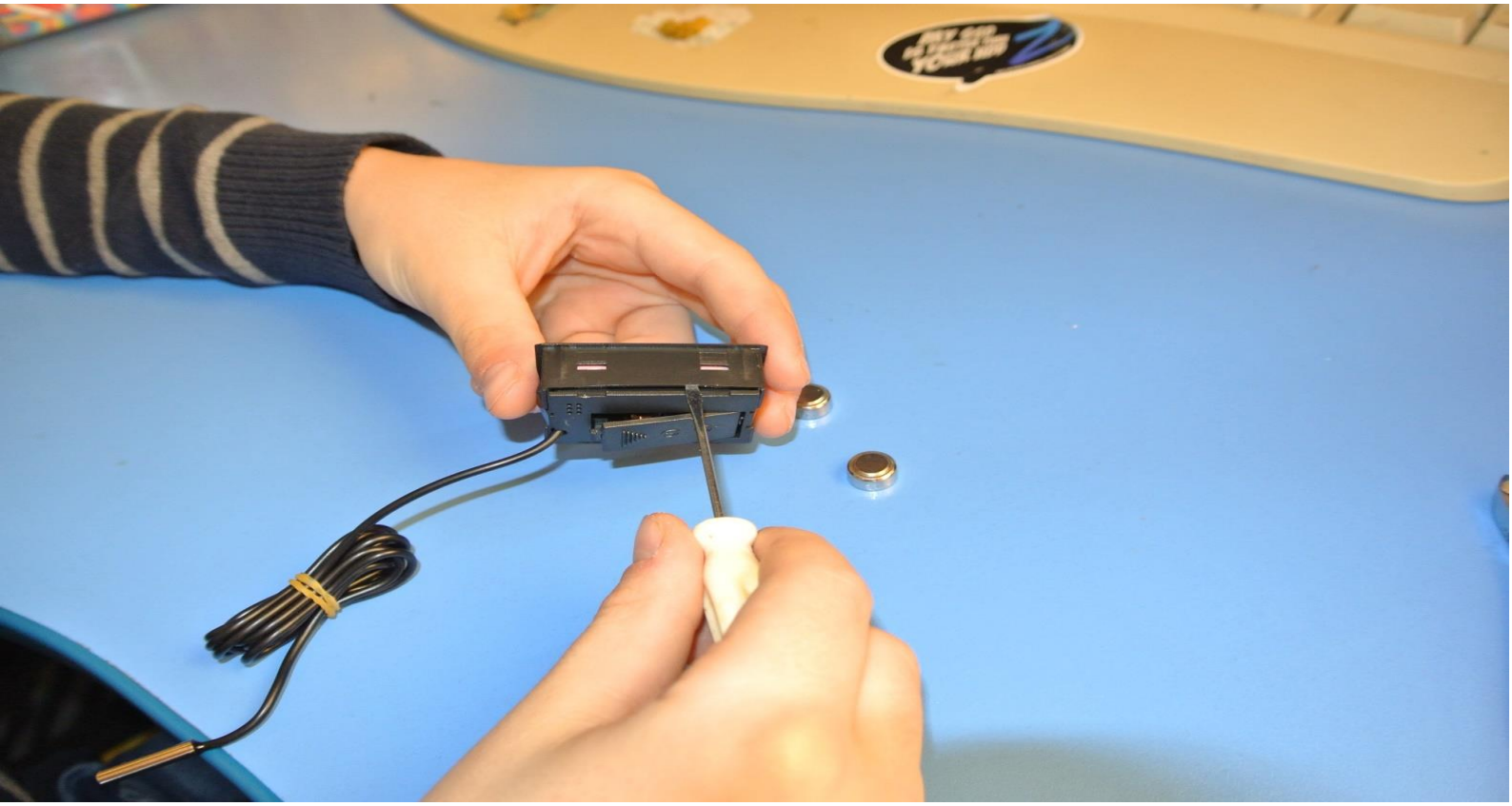


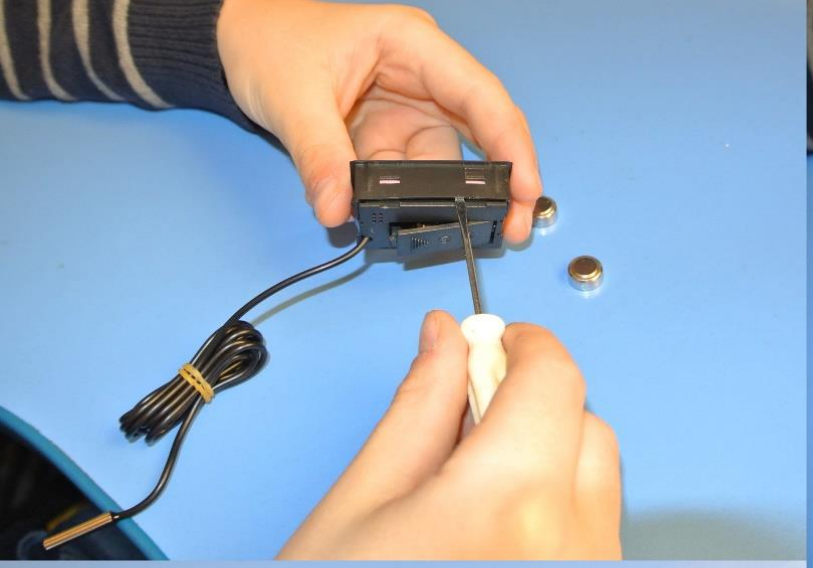
Если научиться нагревать в нем воду до определенной температуры, выдерживая ее определенное время, то поместив внутрь чайника баночку с закваской, мы получим аква-йогуртницу.

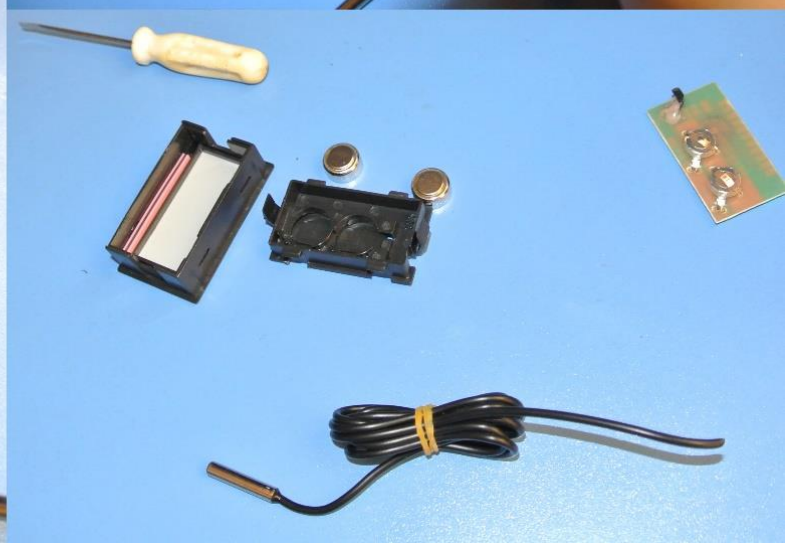
Несомненный плюс этой йогуртницы будет в равномерном распределении тепла вокруг банки с йогуртом, в отличии от классических моделей йогуртниц, в которых нагрев постоянно идет с низу.

Включать и выключать нагрев будет Ардуина в зависимости от температуры воды в чайнике. Для измерения температуры воды в чайнике, Ардуине необходим термодатчик. Под рукою оказался только дешевый электронный термометр с внешним термоэлементом.

Сам термометр нам не нужен, а вот термодатчик пригодится.



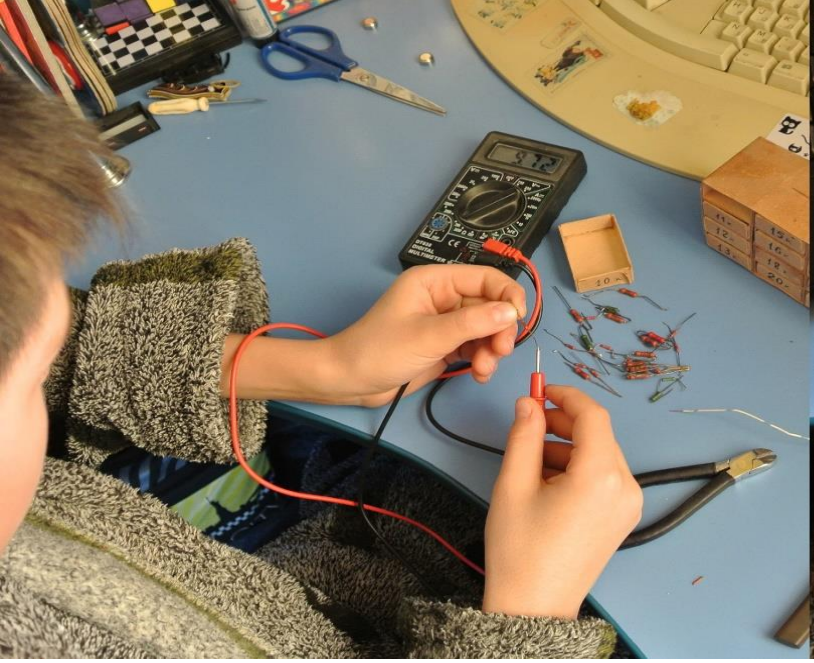


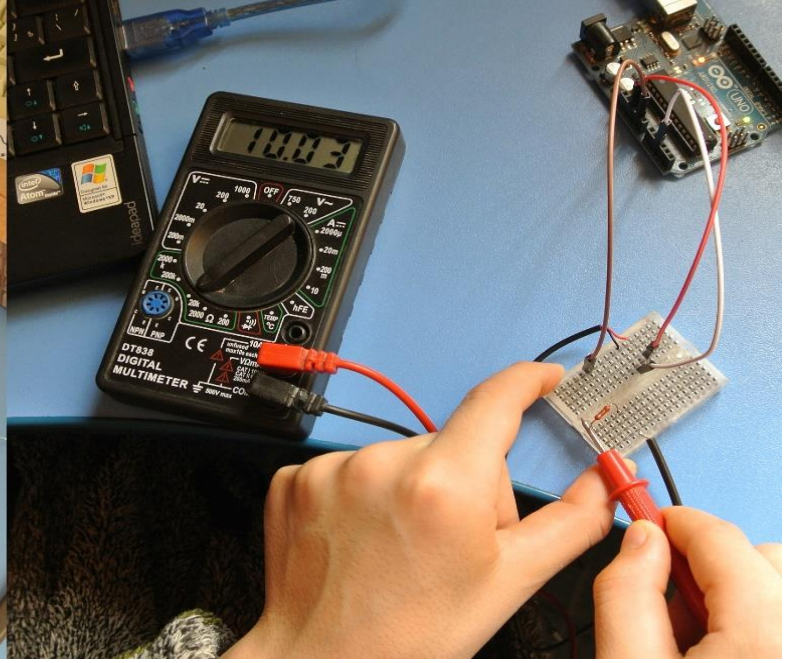



Это оказался самый простой, двух проводной, резистивный датчик.

Как показал поиск в интернете, подключать его надо к аналоговому входу ардуино через простой делитель напряжения.

Сопротивление, используемое для подключения термодатчика должно быть 10 кОм. От его точности будет зависеть правильность показаний термометра. Все стандартные сопротивления имеют не большой разброс параметров. С помощью прибора я нашел сопротивление самое приближенное к 10 кОм.

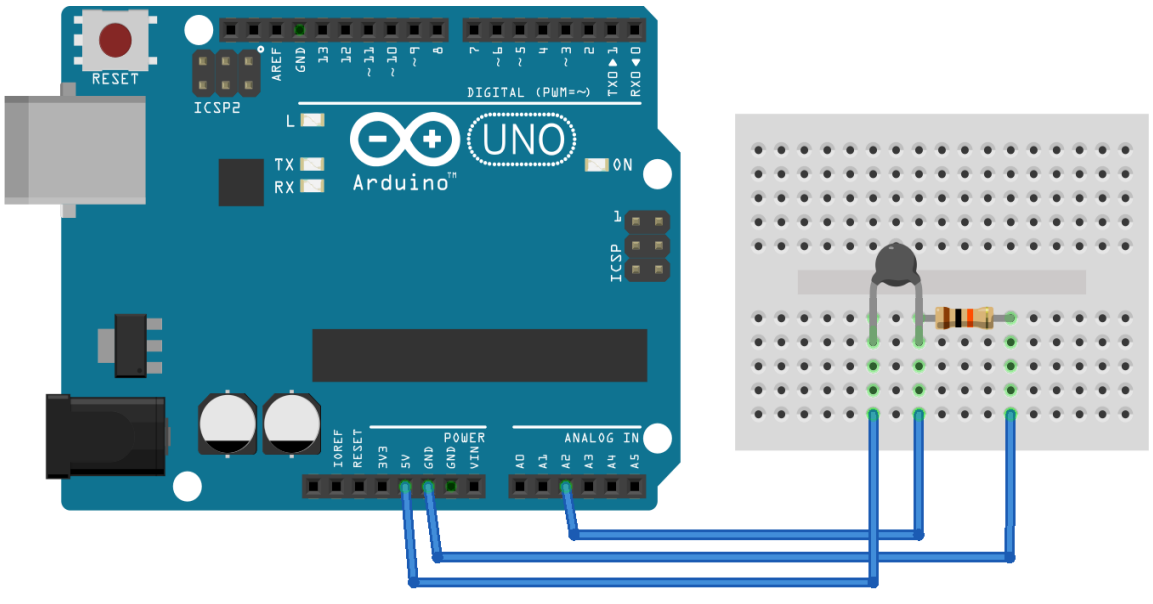




Протестируем температурный датчик, собрав простую схему термометра.

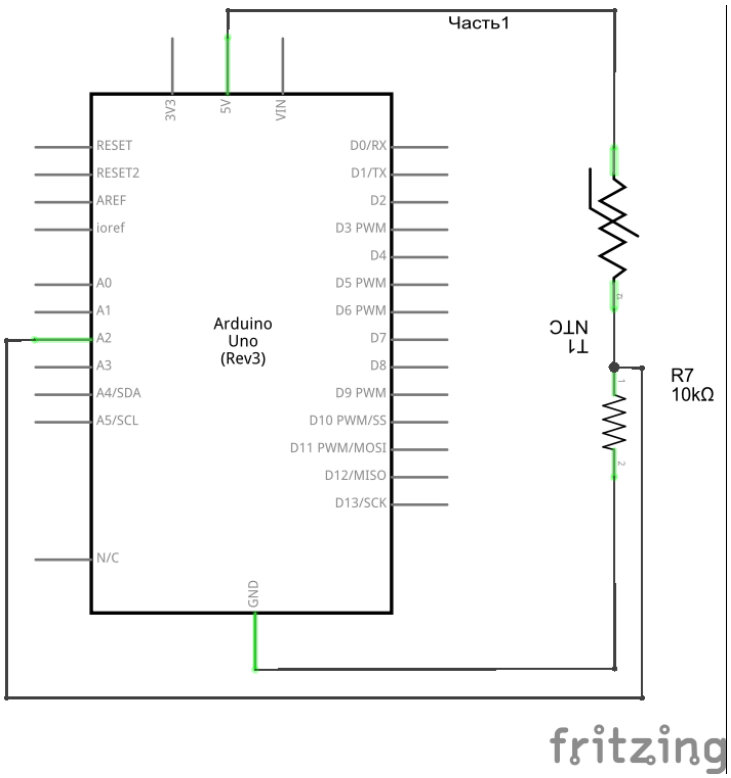
Прежде чем приступить к сборке, нарисуем нашу будущую схему в программе Fritzing. Fritzing создавалась специально для Arduino и имеет набор всевозможных плат с микроконтроллером Atmel AVR, радиодеталей и даже готовых шилдов, имеющихся в продаже. Программа позволяет сделать набросок прототипа в виде рисунка, а также автоматически генерирует принципиальную схему и помогает подготовить печатную плату.

Вот как это выглядит на примере подключения термодатчика:



Картинка соответствует тому, что получается в действительности. Теперь по ней можно очень просто собрать готовую схему.

Принципиальная схема выглядит следующим образом:



Так же в программе Fritzing можно подготовить и печатную плату для ее дальнейшего производства. Но этим мы займемся в самом конце, когда у нас уже будет готов окончательный вариант нашей схемы.

Теперь напишем программу для тестирования термодатчика.

У термодатчика с изменением температуры происходит изменение сопротивления. Однако изменение сопротивления происходит не линейно и прямой зависимости нет. Поэтому просто так перевести значения полученного напряжения на аналоговом входе не получилось. Для этих целей в интернете была найдена формула Стейнхарта и Харта определяющая зависимость сопротивления от температуры:

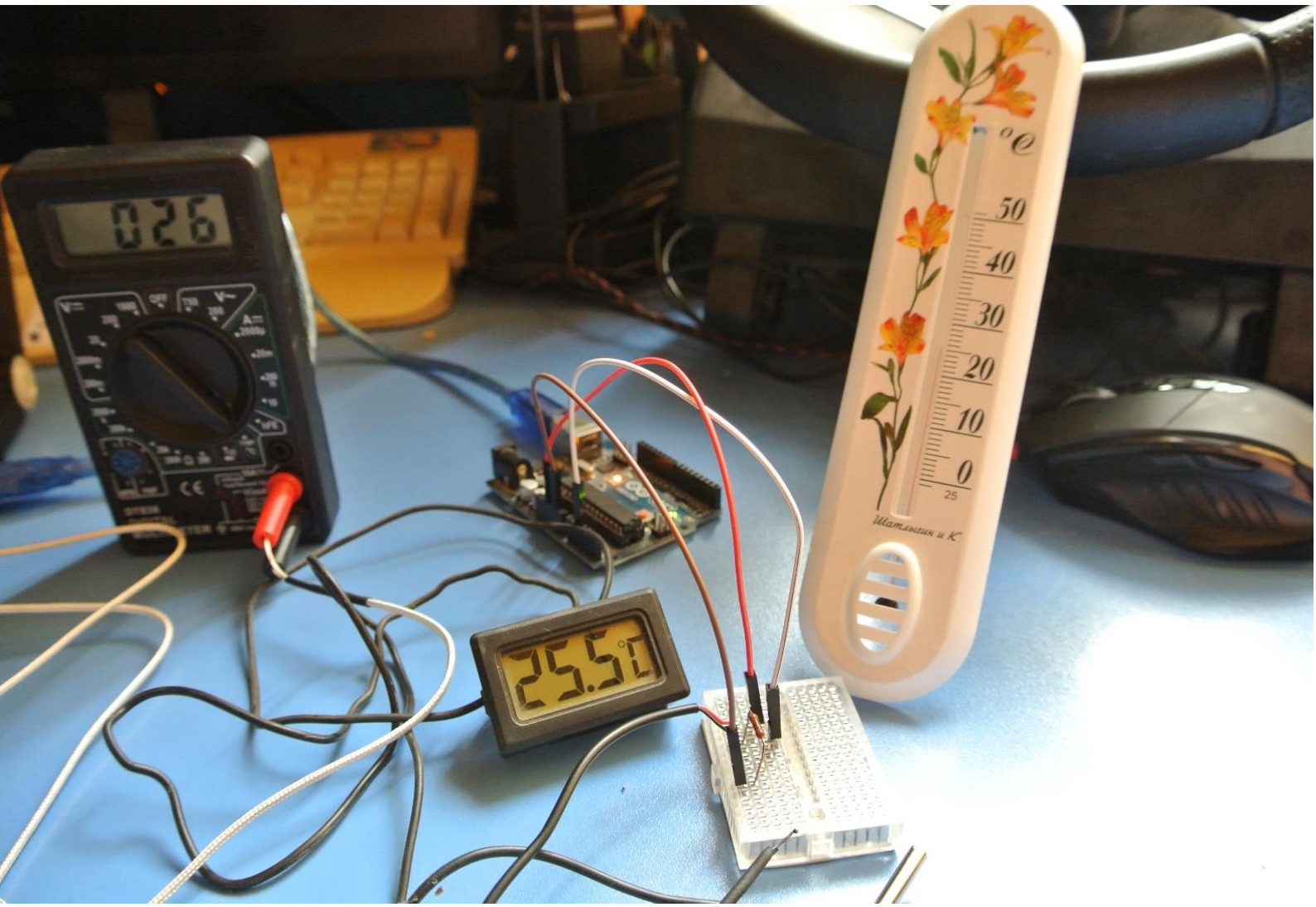
```
1/T = a+b(lnR)+c(lnR)3
```

T — температура в Кельвинах
R — сопротивление
a, b, c — константы термистора, выведенные эмпирически или найденные в документации к термистору

```
#include <math.h>
void setup () {
pinMode(A2,INPUT);
Serial.begin(9600);
}

void loop () {
double temp = analogRead(A2);
temp = log(((10240000/temp) - 10030));
temp = 1 / (0.001129148 + (0.000234125 * temp) + (0.0000000876741 * temp * temp * temp));
temp = temp - 273.15;
Serial.println(temp);

delay(100);
}
```

Константы a,b,c были найдены в интернете для терморезистора имеющего сопротивление 10 кОм при температуре 25гр., как раз такого же как у меня.

Для проверки показаний, выводим данные температуры в порт и сравниваем полученные данные с тремя независимыми источниками



Можно сказать, почти идеально.

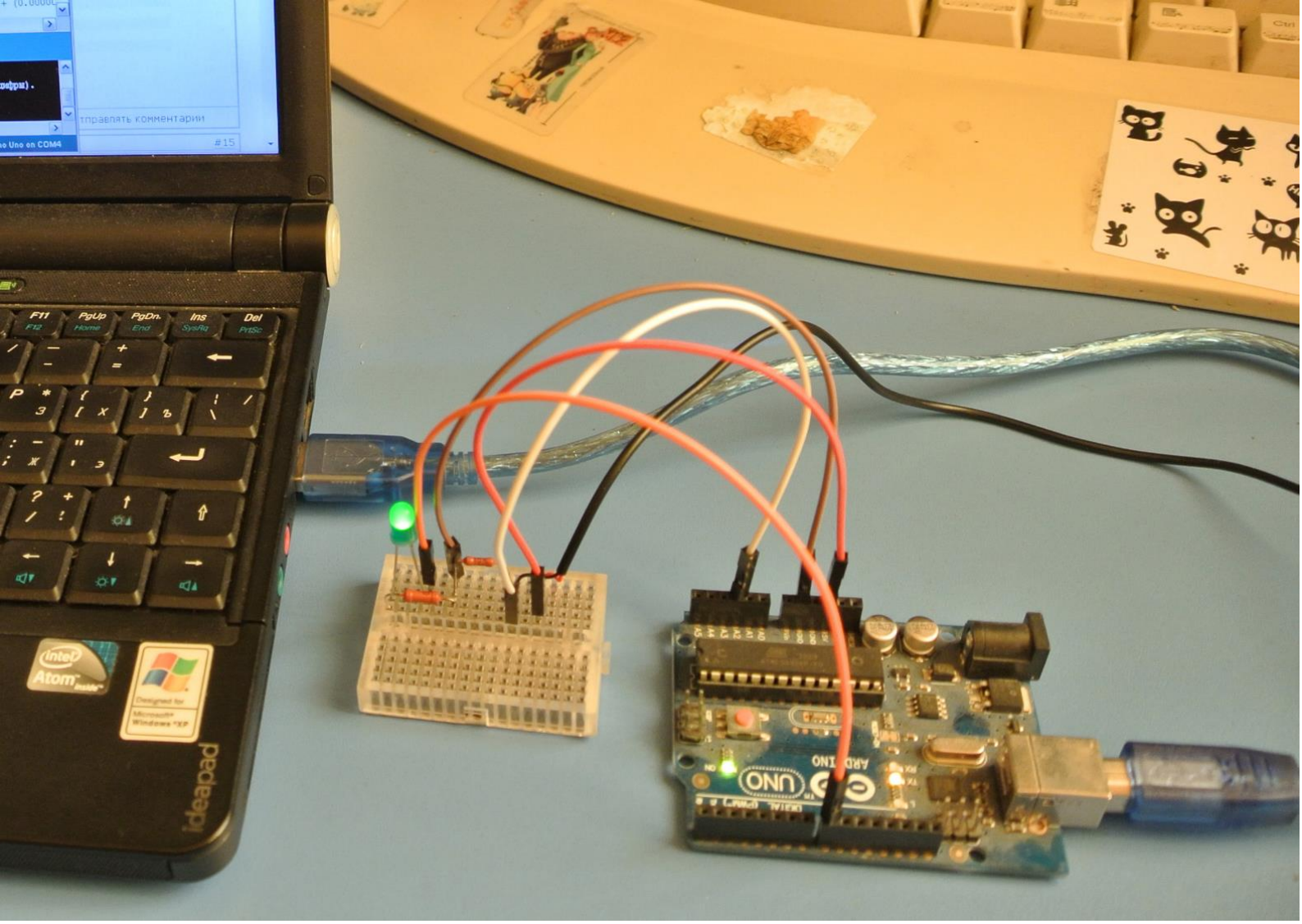
Управлять нагревом чайника должна ардуина, основываясь на показаниях датчика температуры. Но на прямую подключить ардуину к чайнику не получится, чайник работает от 220 вольт. Нужно ставить реле. Но и реле нельзя сразу цеплять на выходы ардуины т.к. выходы у нее очень слабые и на прямую готовы управлять разве что светодиодом. Что бы подключить реле понадобится дополнительная транзисторная обвязка. Можно конечно воспользоваться готовым реле-шилдом, но под рукой его не оказалось. Нашлось решение лучше. Специально для проекта из Китая было заказано твердотельное реле. Это реле полностью электронное, не имеет механических частей (не щелкает) и при этом срабатывает от не большого тока, что позволяет подключать его напрямую к ардуине.

Пока мой заказ шел из Китая, я продолжил создание проекта.

Для тестирования программы я в место реле подключил светодиод и написал не большой скетч, который должен был включать светодиод (в будущем реле) при температуре меньше заданной и выключать, когда температура ее превысит.

```
#include <math.h>
int t = 33; //температура
int led = 8; // pin светодиода

void setup () {
pinMode(A2,INPUT);
pinMode(led, OUTPUT);
}

void loop () {
double temp = analogRead(A2);
  temp = log(((10240000/temp) - 10000));
  temp = 1 / (0.001129148 + (0.000234125 * temp) + (0.0000000876741 * temp *      temp * temp));
  temp = temp - 273.15;

if (temp > t) digitalWrite(led, LOW); // выключаем светодиод

if (temp < t) digitalWrite(led, HIGH); // включаем светодиод

}
```

Но тут меня ждало разочарование. Оказалось, что при наборе температуры показания термометра могут колебаться в верх-низ и порой достаточно быстро. Это отчетливо видно по мерцанию светодиода в момент достижения заданной температуры.



Что бы избежать этого эффекта, пощадить реле и нагревательный элемент программу необходимо сделать таким образом, чтобы она держала включенным светодиод до достижения нужной температуры, после чего светодиод отключался и ждал, когда температура упадет на градус ниже заданного.

Для этого вводим в программу новую переменную, назовем ее *kettle*. Она будет сообщать, идет ли стадия нагрева или стадия остывания.

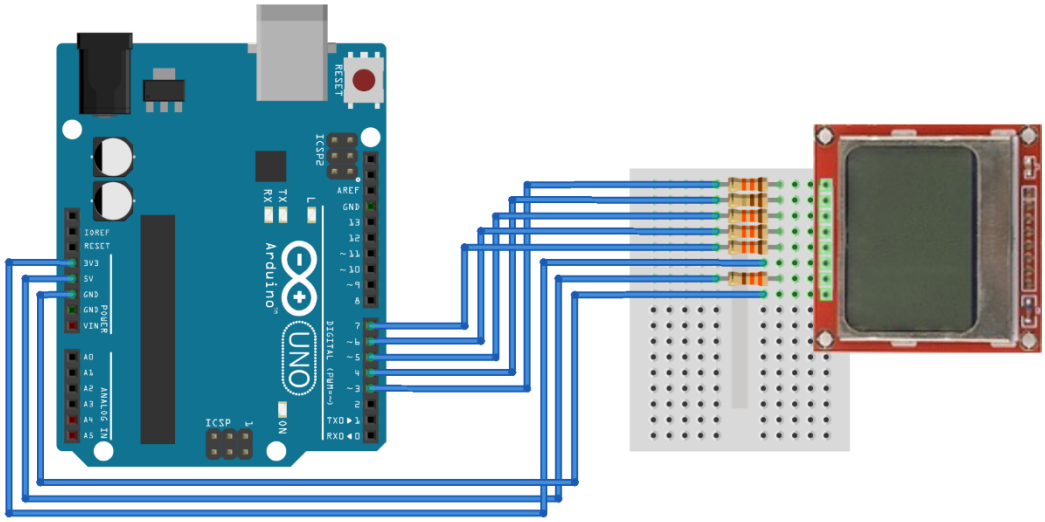
```
#include <math.h>
int t = 33; //температура
int led = 8; // pin светодиода
boolean kettle = true; // true|false либо нагрев либо остывание

void setup () {
pinMode(A2,INPUT);
pinMode(led, OUTPUT);
}

void loop () {
double temp = analogRead(A2);
  temp = log(((10240000/temp) - 10000));
  temp = 1 / (0.001129148 + (0.000234125 * temp) + (0.0000000876741 * temp *      temp * temp));
  temp = temp - 273.15;

// пока идет нагрев – светодиод горит
if (kettle == true) {
  digitalWrite(led, HIGH);
  if (temp > t) kettle = false; // если температура выше заданной вкл. остывание
  };

if (kettle == false) {
  digitalWrite(led, LOW);
  if (temp < t-1) kettle = true; // если температура ниже, включаем нагрев
  };

}
```

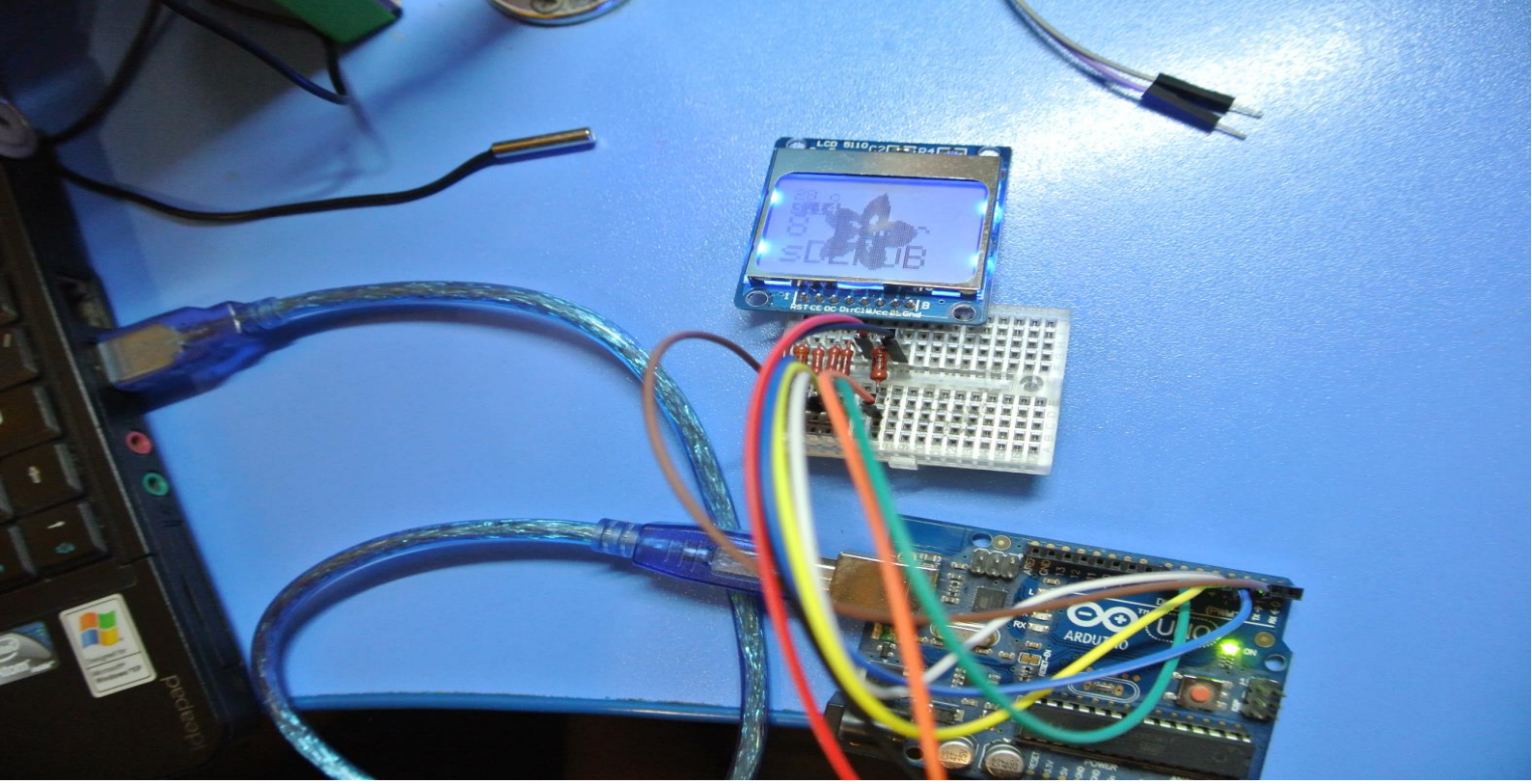
Все получилось отлично, программа заработала так, как надо.

Раз уже мы решили сделать отдельное, не зависящее от компьютера устройство, то нам понадобиться экран для вывода информации. В наличии был экран LCD 5110, который до этого еще не доводилось использовать.

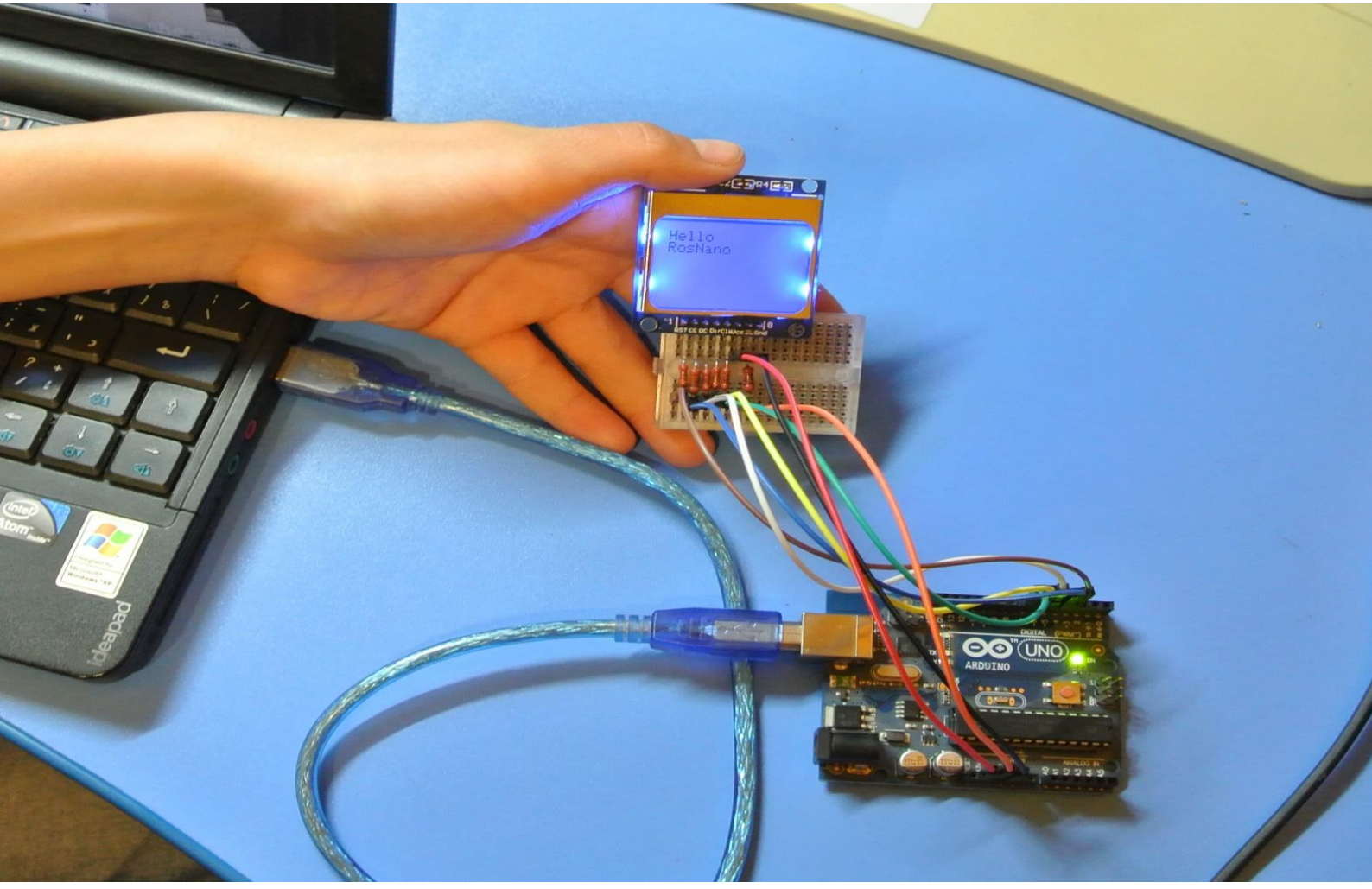
Схема подключения, пробный скетч и необходимая для работы библиотека были найдены в интернете.

Для увеличения срока службы экрана, он был подключен к ардуино через сопротивления.





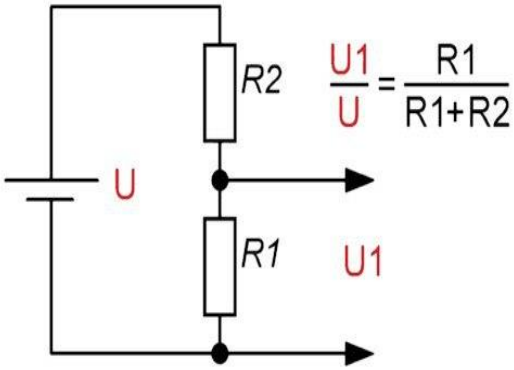
С экраном пришлось повозиться т.к. нормального описания найти не удалось. Но в конечном счете все получилось и теперь можно выводить всю полезную информацию.



Т.к. йогуртница будет программируемой, нужно сделать ввод информации о температуре и времени приготовления.

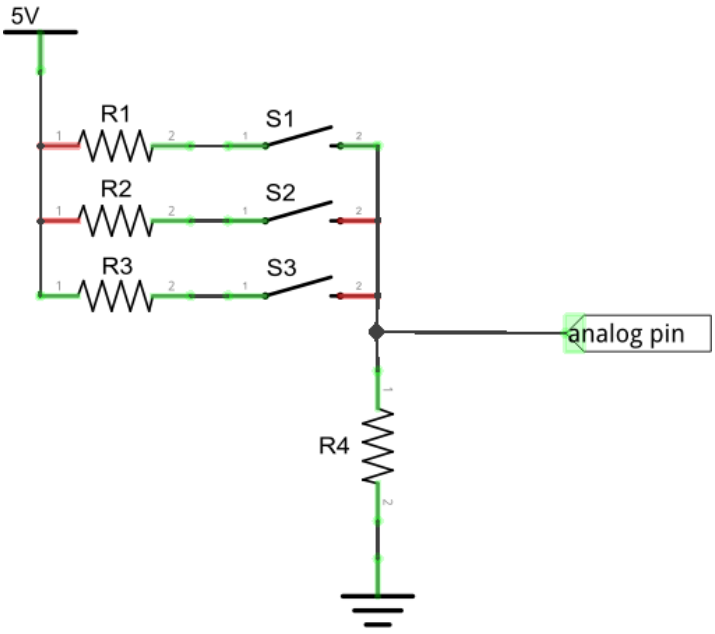
Ввод информации будет осуществляться тремя кнопками, через меню, высвечиваемое на экране ардуины. Кнопки можно подключать разными способами. Самый простой, это подключить каждую кнопку на свой цифровой вход. Но тогда нам потребуется задействовать три цифровых входа. Мне больше понравился вариант подключения всех трех (а можно и гораздо больше) кнопок на аналоговый вход. Тем более такой вариант я еще никогда не пробовал и хотелось поэкспериментировать.

Принцип подключения нескольких кнопок очень прост. Аналоговый вход умеет измерять уровень поступающего на него напряжения. Менять напряжение на входе можно простым делителем напряжения на сопротивлениях



Если оставить сопротивление R1 фиксированным, а сопротивление R2 менять в большую или меньшую сторону, то будет меняться и выходное напряжение.

Поэтому для наших кнопок мы сделаем вот такую схему:



Теперь нажав одну из кнопок, мы задействуем одно из сопротивлений R1, R2 или R3 которое совместно с сопротивлением R4 образует делитель напряжения. Сопротивления R1, R2 и R3 могут быть любыми, но не одинаковыми, а вот сопротивление R4 не должно быть слишком маленьким, иначе может получится большая нагрузка по питанию. Оптимально ставить сопротивление R4 в районе 10 – 20 кОм.

fritzing

Осталось посчитать какое напряжение будет получаться на выходе с учетом тех сопротивлений, которые мы поставили. Можно воспользоваться формулой, но я поступил гораздо проще. Я написал не большую программу в которой сделал вывод данных с аналогового входа, на который подключены кнопки и вывел данные на компьютер через порт.

```
64: void setup() {
65:  pinMode(A1,INPUT);
66:  Serial.begin(9600);
67: }
68: void loop() {
69: Serial.println(analogRead(A1));
70: delay(800);
71: }
```

Нажимая на кнопки, я видел результат, который потом использовал в своей программе.

В моем меню первым должен идти выбор температуры, при которой будет делаться йогурт. Для изменения температуры задействованы две кнопки. Одна увеличивает, а другая уменьшает значение температуры. При этом, чтобы долго не «проматывать» кнопками выбирая нужное значение, при включении появляется предустановленное значение температуры, очень близкое к часто используемым. Так же точно сделано в меню выбора времени. Только в меню времени, при нажатии на кнопку, оно увеличивается не на одну минуту, а сразу с шагом в 5 минут. Для производства йогурта это не критично, зато позволяет быстрее выбрать нужную температуру.

Третья кнопка – это кнопка Enter, для подтверждения введенных данных. После введенного времени по кнопке enter йогуртница начинает процесс приготовления.

Для организации вывода меню на экран мне понадобилось прибегнуть к одной хитрости. В *viod setup* я поместил бесконечный цикл, в котором постоянно шел опрос моих кнопок. Нажатие первых двух кнопок приводило к изменению значений с последующим отображением на экране, а нажатие кнопки enter завершало цикл.

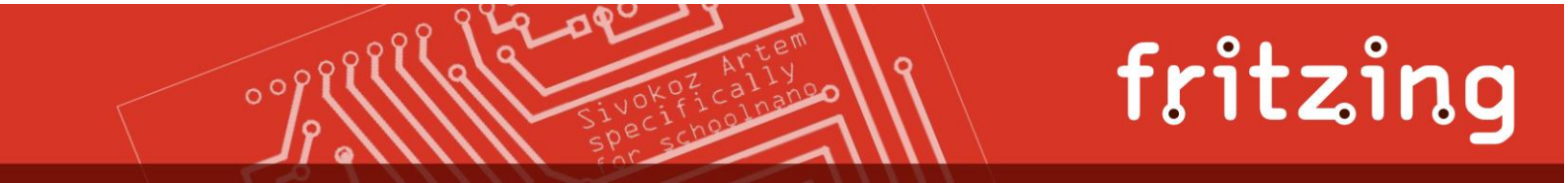
Пример кода:

```
void setup()   {
x=0;
do {
        pinA3 = analogRead(A3);
        Serial.println(analogRead(A3));
        if (pinA3 > 700 && pinA3 < 1100) //если нажата кнопка вверх
                 {
                  temperature ++;
                  delay(200);
                 }
        if (pinA3 > 450 && pinA3 < 600) //если нажата кнопка вниз
                 {
                  temperature --;
                  delay(200);
                 }
        if (pinA3 > 250 && pinA3 < 400) //если нажата кнопка enter
                 {
                  X = 1;
                 }
    }  while (x=1);
```

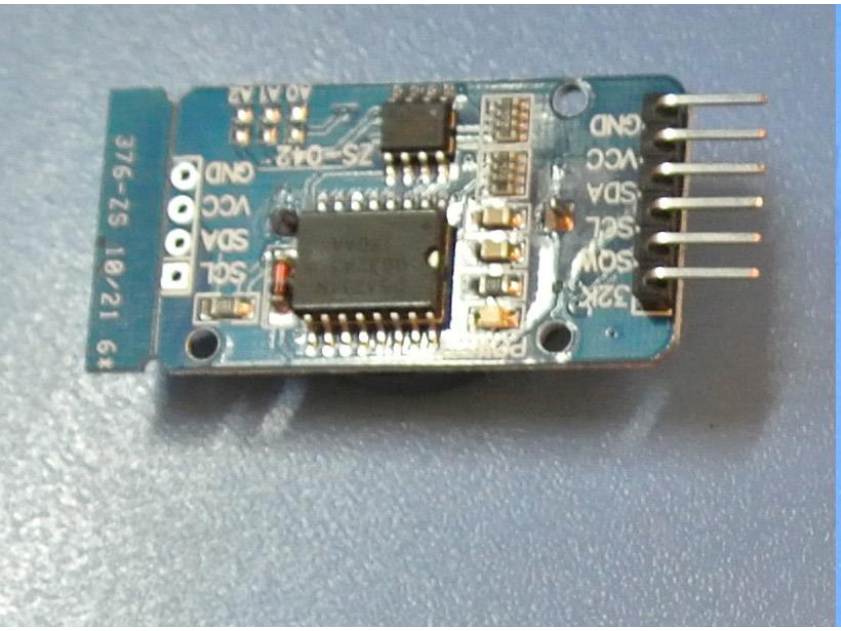
Но т.к. кнопка enter имела самый маленький диапазон значений, то иногда случались ложные срабатывания при нажатии других кнопок.
Избавиться от ложного срабатывания кнопки получилось путем нескольких опросов подряд кнопки, для того что бы удостовериться действительно ли она была нажата.
Для этого переменная X не сразу равнялась 1, а несколько раз увеличивалась X++, пока не достигала значения 5 и только тогда цикл завершался. Если срабатывание было ложным, то нажатие любой другой кнопки сбрасывало X на 0. На глаз это не заметно, зато от ложных срабатываний помогло избавиться.
Пример кода:

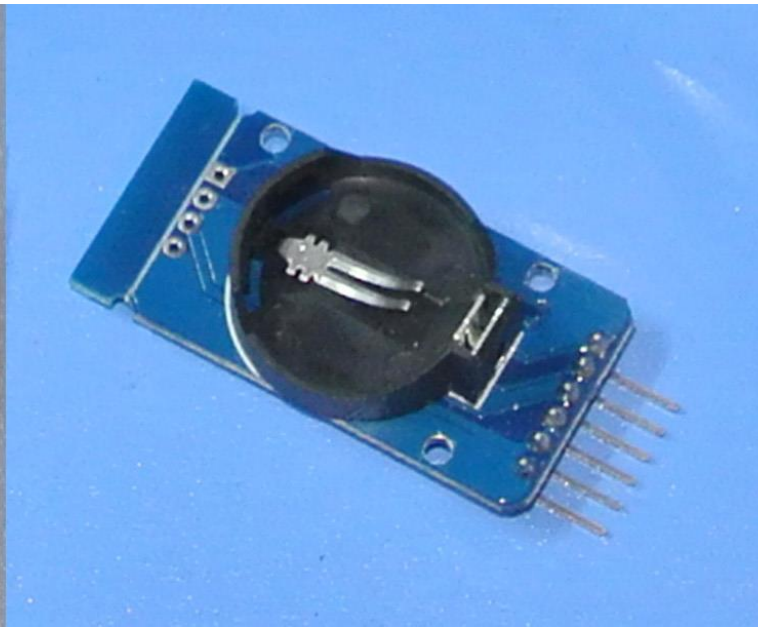
```
void setup()   {
x=0;
do {
        pinA3 = analogRead(A3);
        Serial.println(analogRead(A3));
        if (pinA3 > 700 && pinA3 < 1100) //если нажата кнопка вверх
                 {
                  temperature ++;
                  x=0;
                  delay(200);
                 }
        if (pinA3 > 450 && pinA3 < 600) //если нажата кнопка вниз
                 {
                  temperature --;
                  x=0;
                  delay(200);
                 }
        if (pinA3 > 250 && pinA3 < 400) //если нажата кнопка enter
                 {
                  x++;
                 }
    }  while (x<5);
```

Практически такой же код сделан и для меню выбора времени работы.



Теперь нужно организовать цикл работы йогуртницы равный заданному в меню времени. Но тут я столкнулся с одной проблемой. Проверка показала, что отслеживать время приготовления программным способом (уменьшение заданной переменной за каждый цикл) не получается т.к. количество команд в программе может меняться, а соответственно и скорость прохождения каждого цикла может быть разной. Для простейшей йогуртницы погрешность в полчаса была бы не критичной, но т.к. мы разрабатываем не просто йогуртницу, а целую лабораторию, то точность нам нужна. Для точного отслеживания времени нам понадобятся «часы реального времени».





Часы подключаются очень просто: вывод GND подключаем к «земле», вывод VCC — к напряжению питания ардуино. А провода для синхронизации и данных в Arduino UNO подключаются к входам A4 (SDA) и A5 (SCL).

Если в модуль поставить батарейку, то часы будут работать даже после отключения внешнего питания. Но нам это не нужно т.к. я не буду записывать в модуль точное время, а буду его использовать исключительно как секундомер.

Для этого потребуется подключить универсальную библиотеку для часов реального времени DS1302, DS1307, DS3231

```
114:  #include <RTC.h>
```

И в момент начала работы йогуртницы обнулить время в таймере

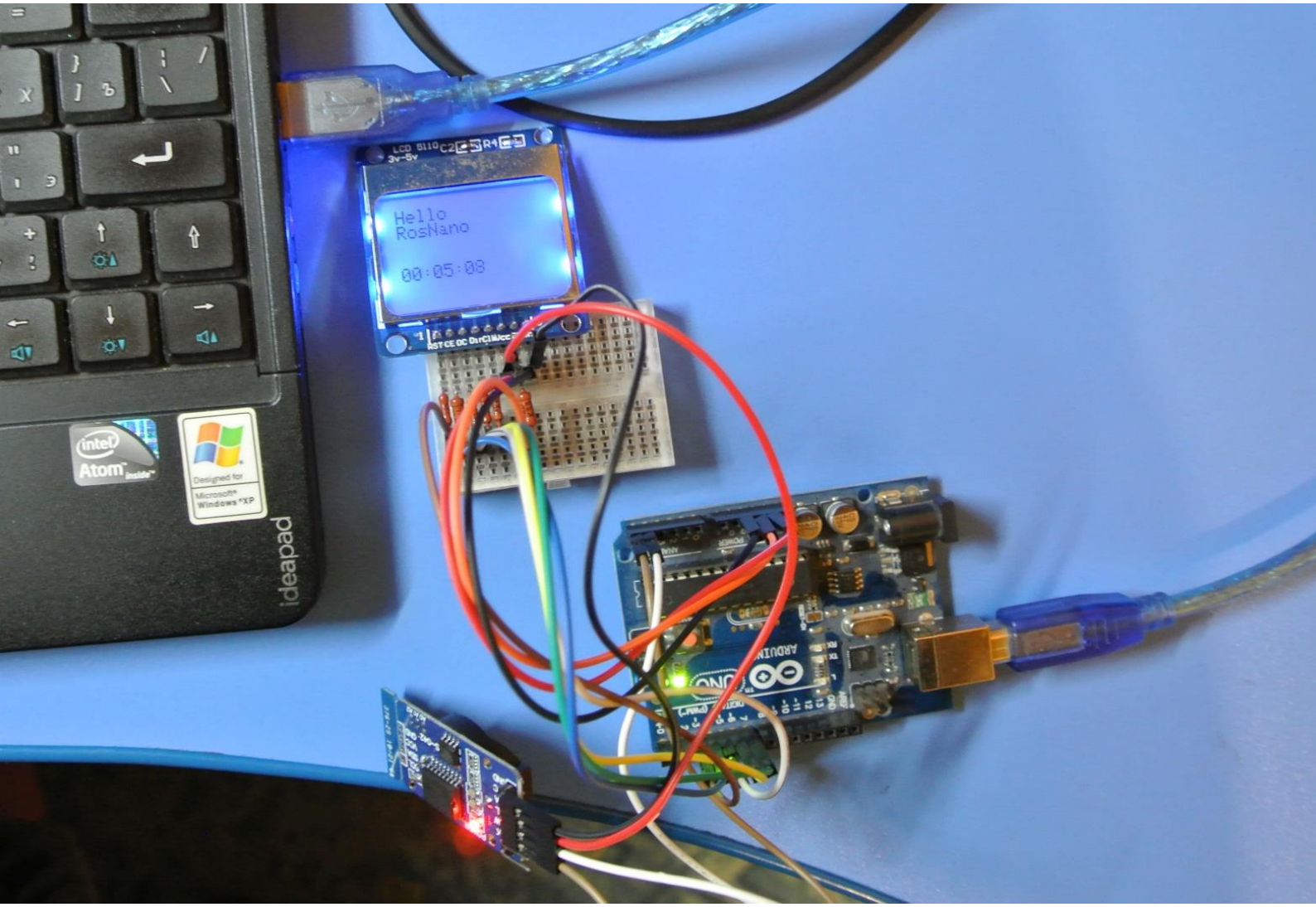
```
115:  time.settime(0,0,0,0,0,0,0); //устанавливаем время в таймере
```

В конце цикла делаем проверку, если время вышло, то выход из цикла приготовления

```
116:  } while ((time.Hours * 60 + time.minutes) < timer); //если время вышло - завершаем
```

В процессе приготовления можно так же опрашивать таймер и выводит текущее время. Либо путем простейшей арифметической операции показывать оставшееся для приготовления время.

```
117:  display.println(timer - (time.Hours * 60 + time.minutes));
```

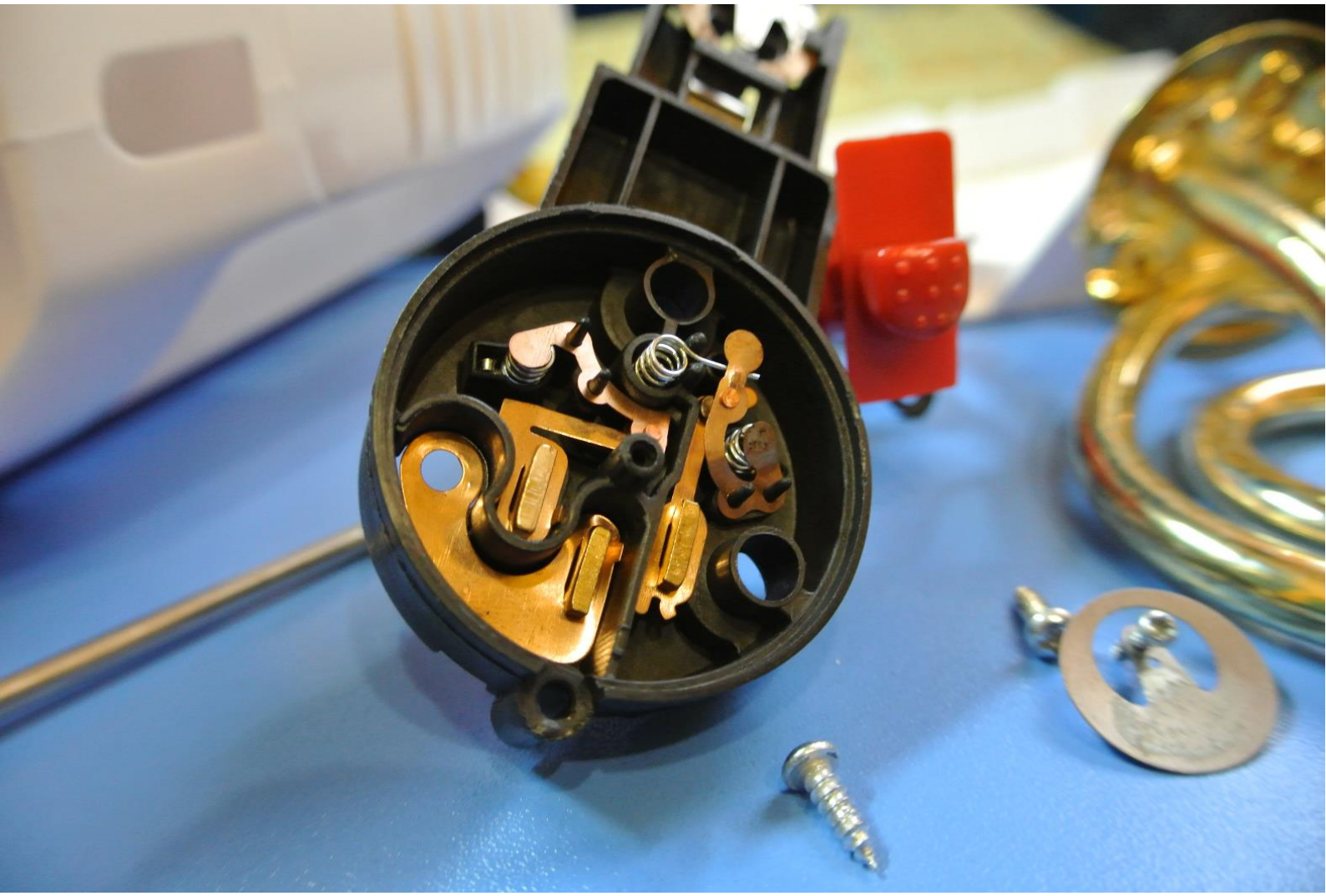


Пока твердотельное реле было еще в пути, необходимо было придумать способ его подключения к чайнику, а также создание автономного питания ардуины.

Самый простой способ управления чайником, сделать отдельную розетку, управляемую реле, для подключения в нее чайника. Но в таком случае нам придется делать отдельно кнопку включения для ардуины, а также тянуть провода от датчика температуры находящегося в чайнике. Но т.к. твердотельное реле было еще в пути и было немного свободного времени, решено было изучить устройство чайника для подключения управления напрямую.

Разбирается чайник очень легко, достаточно выкрутить несколько шурупов. Вынимаем тэн и черную коробочку управления тэном.

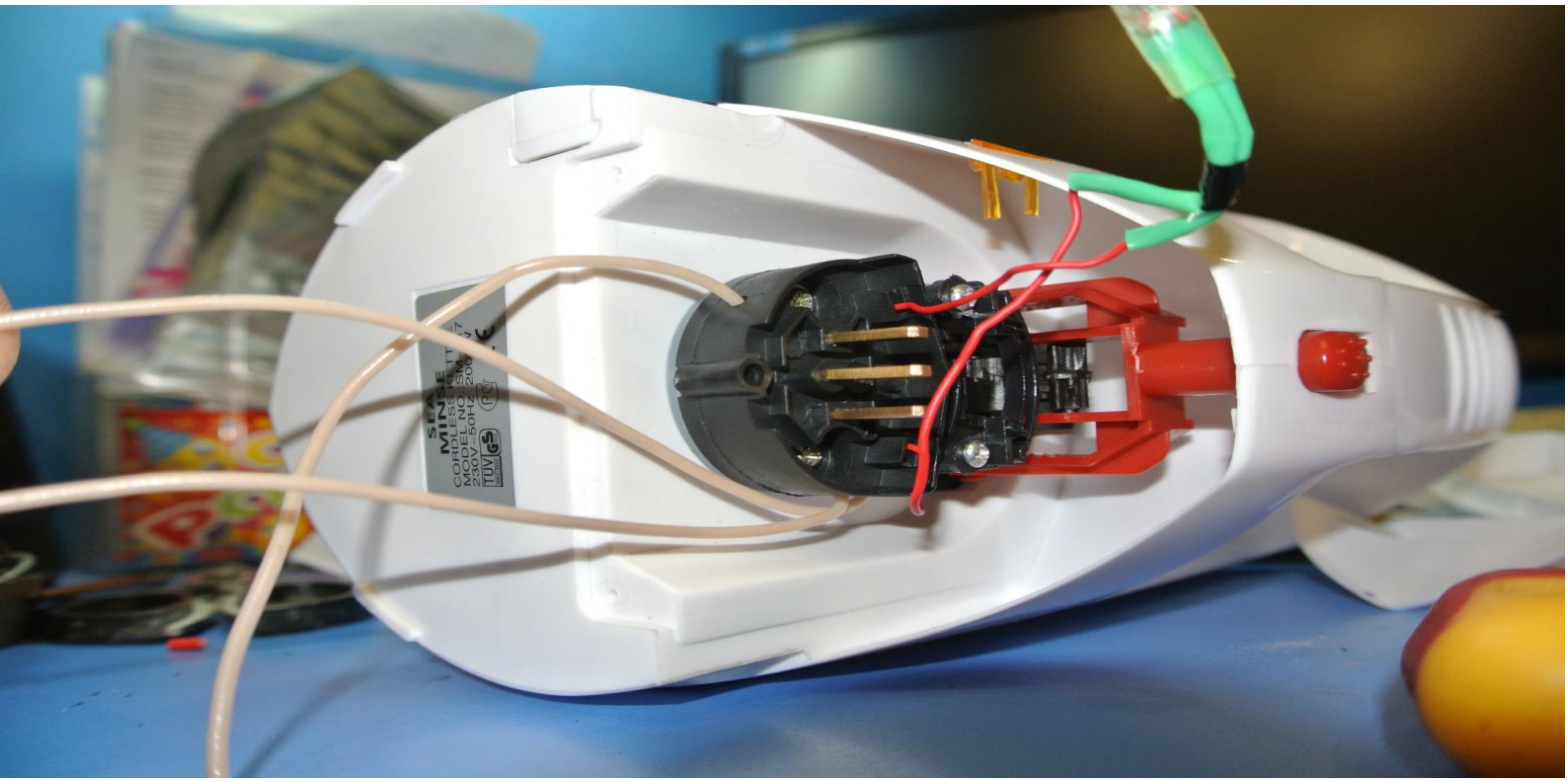
В нутрии черной коробочки к которой подключена кнопка включения/отключения находятся контакты, которые управляют работой чайника.



В конструкцию были внесены не большие изменения.

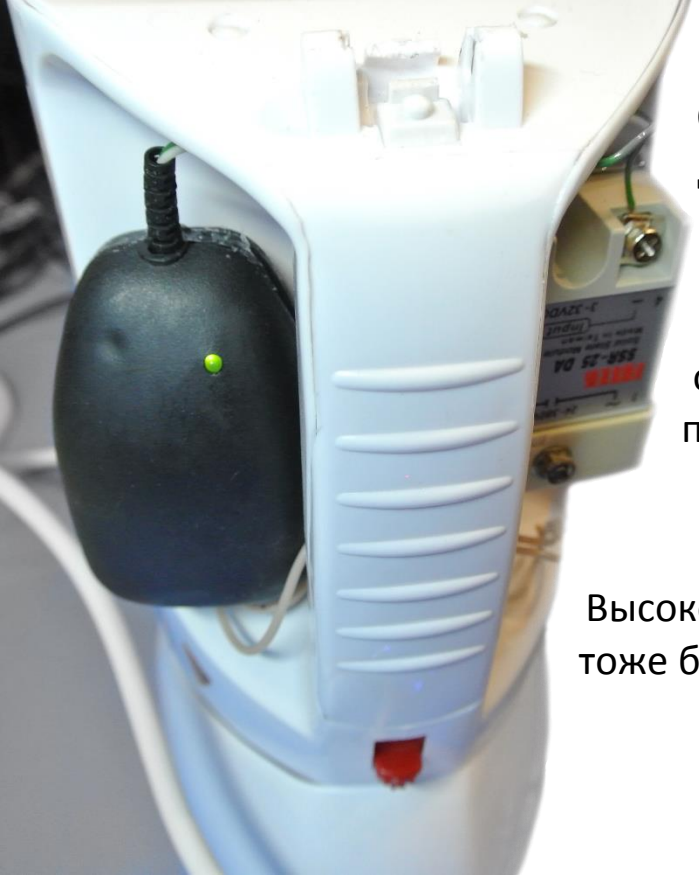
Провода были подключены таким образом, чтобы стандартная кнопка включения чайника подавала напряжение на блок питания ардуины, но сам нагревательный элемент был отключен и включался только когда будут замкнуты другие провода, подключенные к твердотельному реле.

Такое подключение позволяет управлять включением йогуртницы только одной кнопкой, а также в случае зависания ардуины или сбоя в работе программы, когда реле зависает во включенном состоянии, в момент закипания воды сработает штатная кнопка и обесточит полностью все устройство.



К тому моменту как был доработан чайник, по почте пришло твердотельное реле. Выглядит оно вот так:

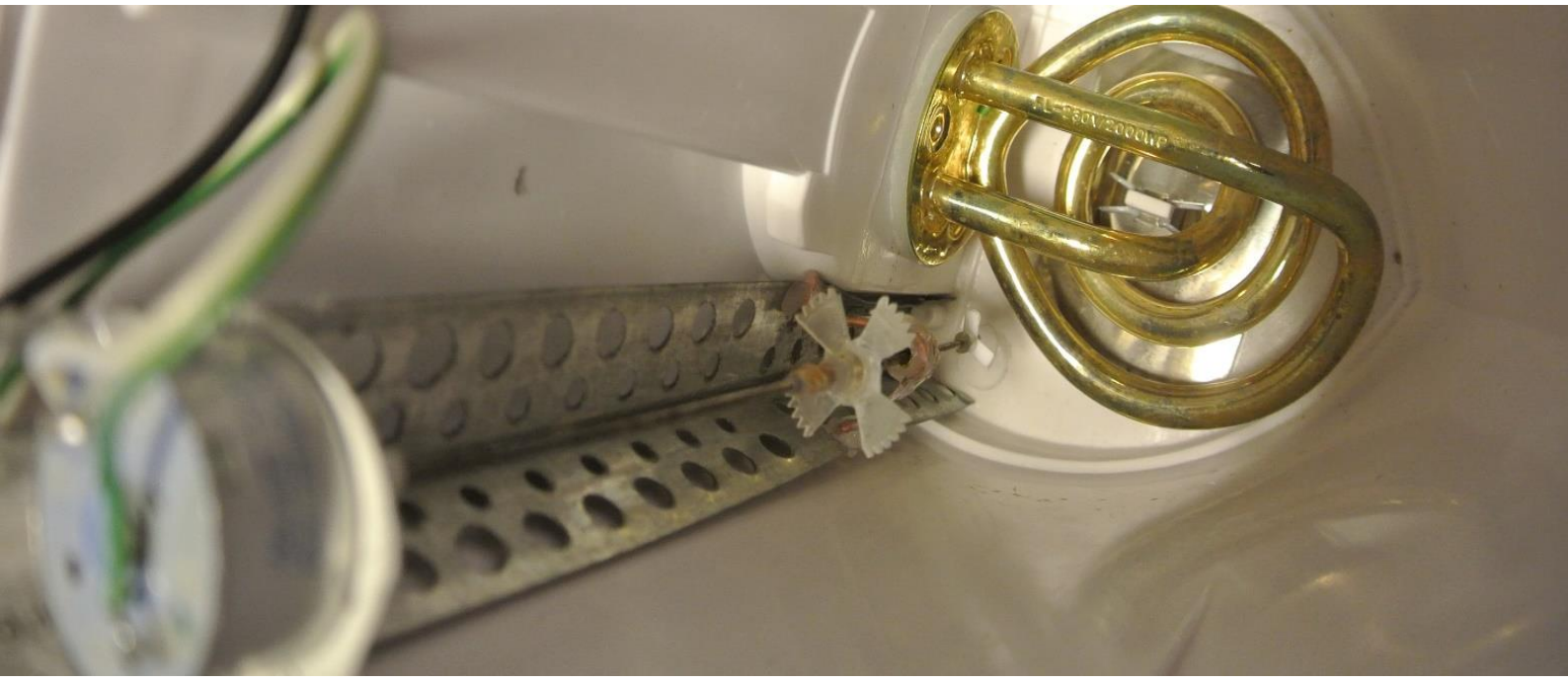
Для питания ардуины нужен внешний блок питания. Применяемая в проекте Arduino UNO может работать от внешнего источника питания напряжением от 6 до 20 Вольт. Рекомендуемые значения, указанные в описание от 7 до 12 Вольт. Поэтому для питания был взят готовый БП от внешней телевизионной антенны.



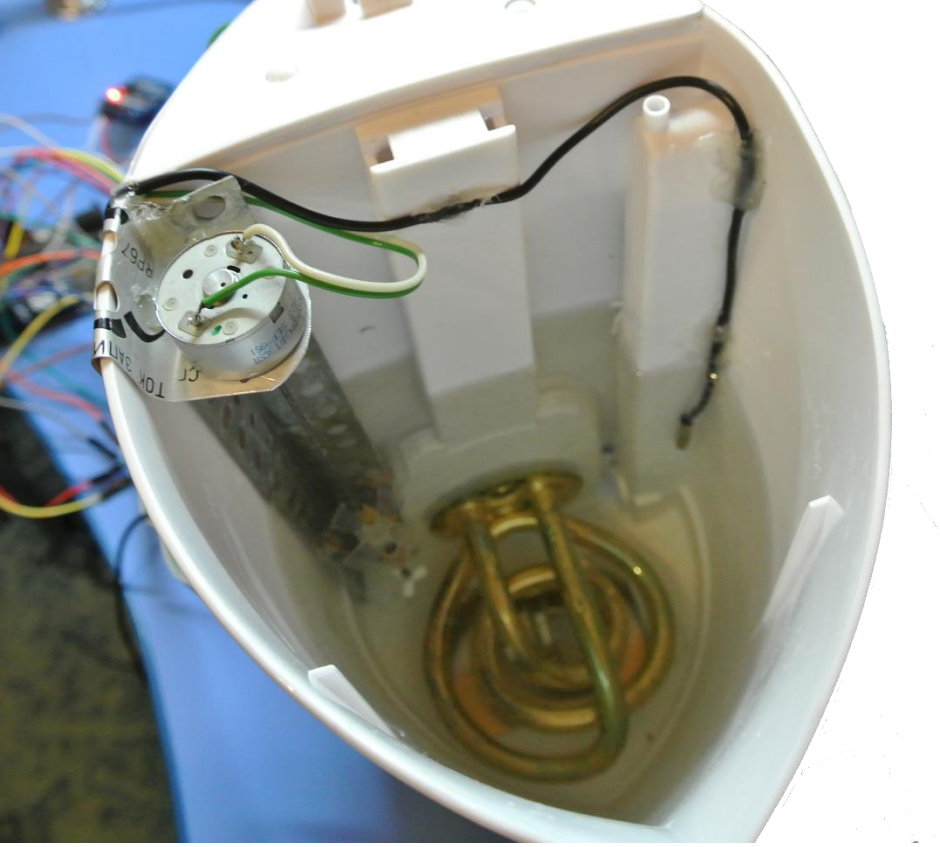
Он подвергся не большой дополнительной доработке, была обрезана вилка и выведены провода напрямую из корпуса, для того что бы не было открытых токоведущих частей высокого напряжения, опасных для жизни. А также можно было его приклеить к корпусу чайника.

Высоковольтные контакты твердотельного реле тоже были закрыты пластиковой крышечкой.

Что бы нагрев воды от спирали чайника был равномерным, а также для более точного замера температуры, в конструкцию был добавлен двигатель, перемешивающий воду в момент ее нагрева.

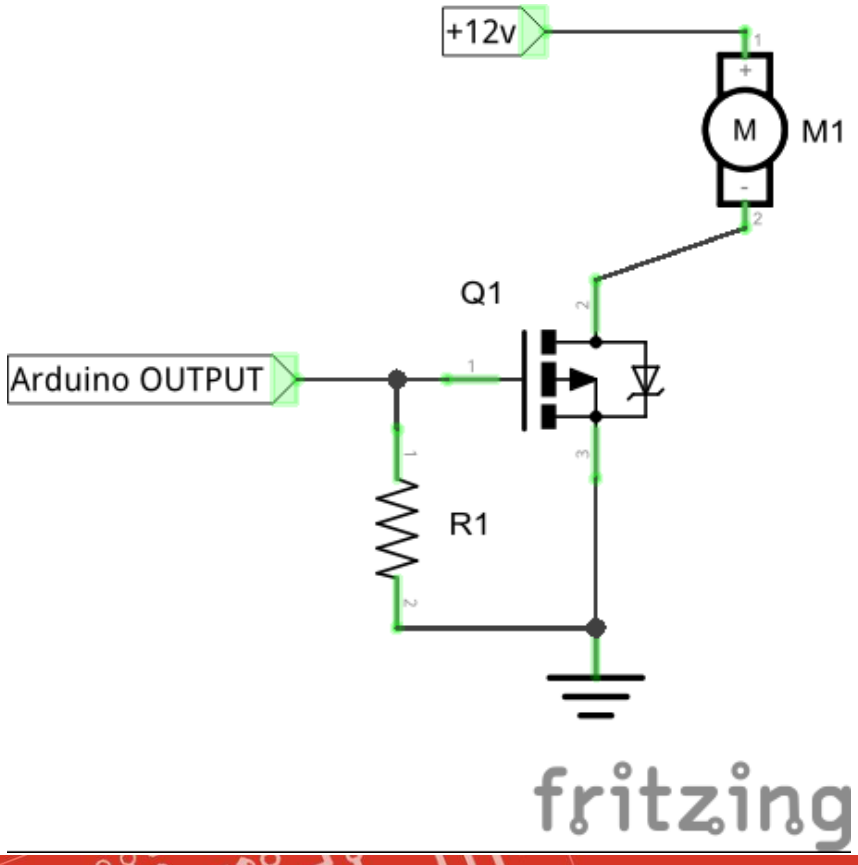


Конструкция правда несколько раз переделывалась пока был достигнут оптимальный вариант



Электродвигатель для ардуино оказался не посильной нагрузкой и подключить его на прямую не получилось, но помог прошлый опыт создания кораблика, в котором коллекторный двигатель управлялся полевым транзистором с диодом шотки.

Получилась вот такая, очень простая схема.

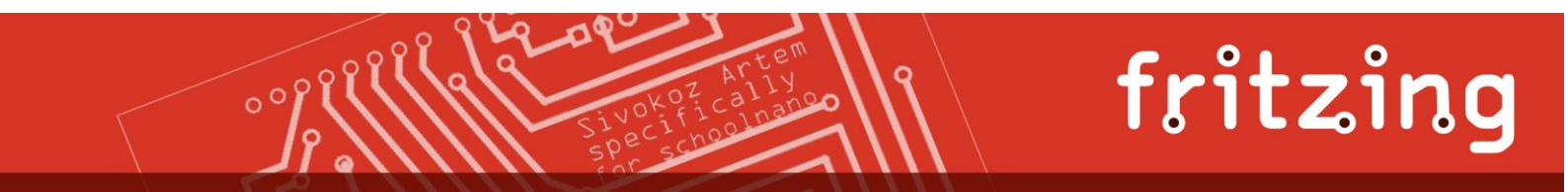


Двигатель заработал от ардуины, но работал он не долго, примерно около 30 секунд, после чего, его обороты стремительно падали, он останавливался и отключалась ардуина. Через некоторое время он снова начинал работать, но также примерно с пол минуты. В чем была причина помог понять тестер, подключенный к блоку питания. Оказалось, что в блоке питания от антенного усилителя установлен очень слабенький стабилизатор, который при большой нагрузке сразу же начинал очень сильно греться и выходное напряжение падало до нуля.

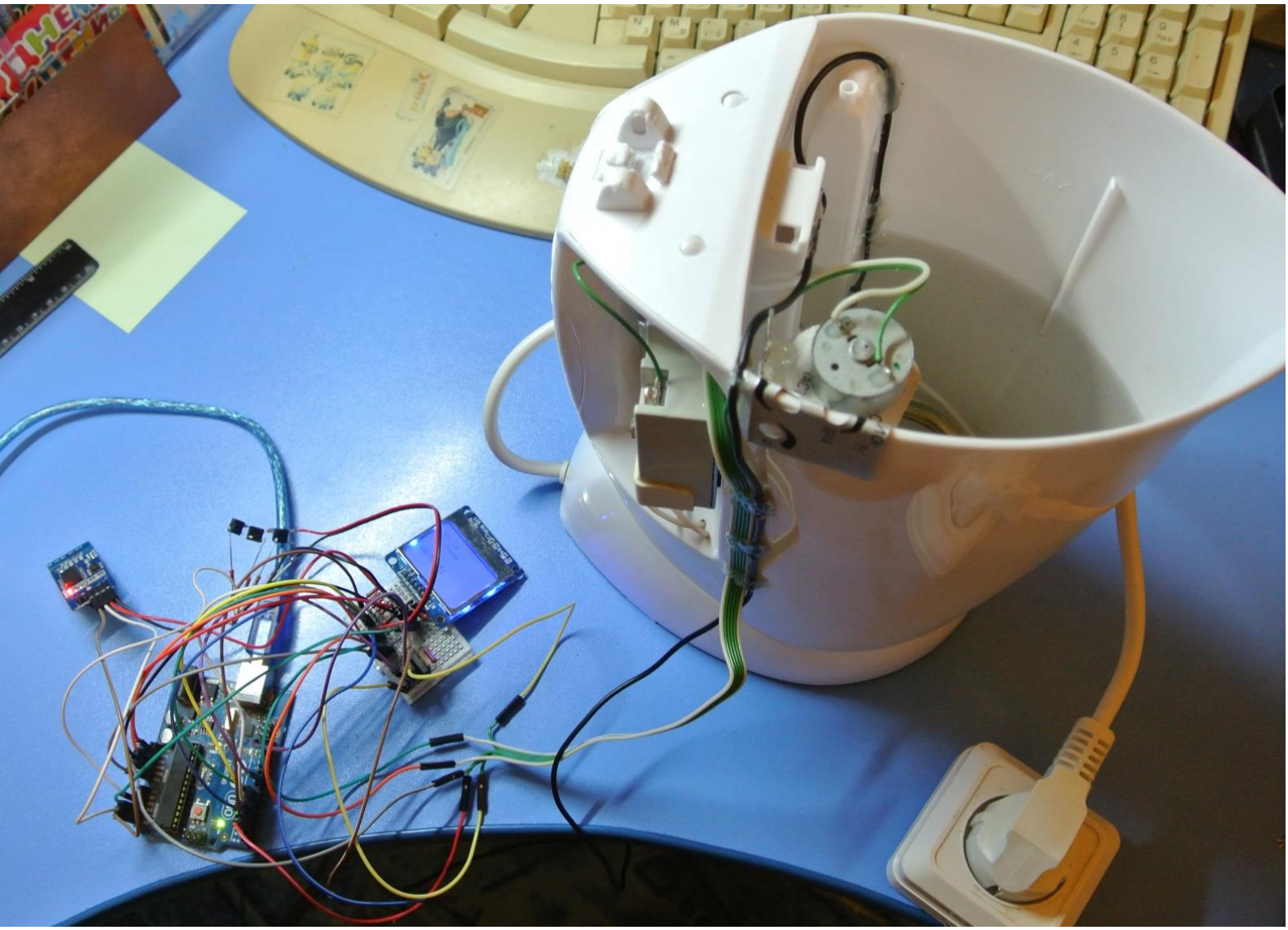
Блок питания пришлось переделать. В место установленного стабилизатора был впаян более мощный стабилизатор на 12 вольт КР142ЕН8Б. Двигатель заработал в полную мощность. Обороты двигателя оказались слишком большими, их надо было уменьшить. Самый простой способ уменьшить обороты коллекторного двигателя – подать на него меньшее напряжение питания. Но подавать на него питание не 12 вольт, а 5 от ардуины не желательно, можно спалить ардуину. Ставить сопротивление тоже не самый лучший способ. Мне опять помог мой опыт создания кораблика, в котором управление «газом» двигателя осуществлялось ШИМ сигналом. В Arduino UNO есть несколько выходов, которые можно очень простым способом задействовать под ШИМ. Делается это очень просто, вот пример кода:

```
118: #define moto 9
119:
120: void setup() {
121: pinMode(moto, OUTPUT);
122: }
123:
124: void loop() {
125: analogWrite(moto, 200);
126: }
```

В зависимости от поставленного значения в 79й строке, будет изменяться скорость вращения двигателя.



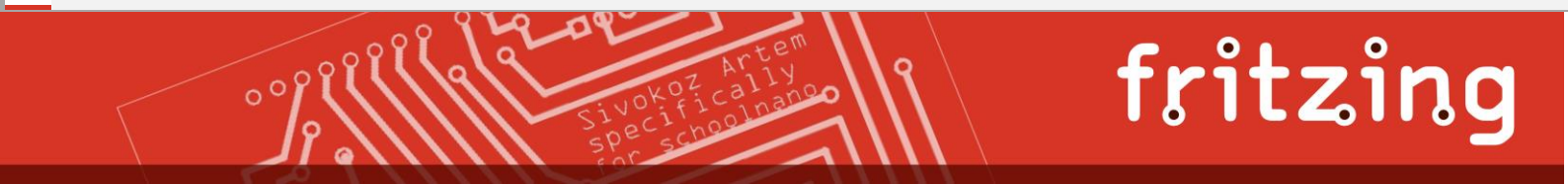
Теперь осталось все разом подключить.



И написать программу управления:

```
#include <math.h> // библиотека для выполнения простых математических операций
#include <RTC.h> //универсальная библиотека для часов реального времени DS1302, DS1307, DS3231
#include <Adafruit_GFX.h> //библиотека для графических дисплеев
#include <Adafruit_PCD8544.h> //для NOKIA LCD 5110

#define moto 9
#define rele 2

int x = 0;
int pinA3 = 0;
int temperature = 36;
int timer = 360;
boolean kettle = true;

RTC time;
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

void setup()   {
 Serial.begin(9600);
 pinMode(A3,INPUT);
 pinMode(moto,OUTPUT);
 pinMode(rele,OUTPUT);
 time.begin(RTC_DS3231);

 display.begin();
 display.clearDisplay();
 display.setContrast(55);
 display.setTextColor(BLACK);

do {
       pinA3 = analogRead(A3);
       Serial.println(analogRead(A3));
       if (pinA3 > 700 && pinA3 < 1100)
                {
                 temperature ++;
                  x=0;
                  delay(200);
                }
       if (pinA3 > 450 && pinA3 < 600)
                {
                 temperature --;
                  x=0;
                  delay(200);
                }
       if (pinA3 > 250 && pinA3 < 400)
                {
                 x++;
                }
       display.clearDisplay();
       display.setCursor(0,0);
       display.setTextSize(1);
       display.println("Temperature:");
       display.setCursor(0,20);
       display.setTextSize(3);
       display.println(temperature);
       display.display();
   }  while (x<5);

x=0;
delay(200);

do {
       pinA3 = analogRead(A3);
       Serial.println(analogRead(A3));
       if (pinA3 > 700 && pinA3 < 1100)
                {
                 timer = timer + 5;
                  delay(200);
                  x=0;
                }
       if (pinA3 > 450 && pinA3 < 600)

                {
```

```
                  timer = timer - 5;
                   delay(200);
                   x=0;
                 }
        if (pinA3 > 250 && pinA3 < 400)
                 {
                  x++;
                 }
        display.clearDisplay();
        display.setCursor(0,0);
        display.setTextSize(1);
        display.println("Time:");
        display.setCursor(0,20);
        display.setTextSize(3);
        display.println(timer);
        display.display();
    }  while (x<5);

time.settime(0,0,0,0,0,0,0); //устанавливаем время в таймере
}


void loop() {

do {
display.clearDisplay();
double temp = analogRead(A0);

  temp = log(((10240000/temp) - 10000));
  temp = 1 / (0.002229327435341785 + (0.0002401007279668999 * temp) + (0.000000020839230886815 97 * temp * temp * temp));
  temp = temp - 273.15;

//греем чайник
if (kettle == true) {
  analogWrite(moto, 200); //включаем мотор
  digitalWrite(rele, HIGH);

  display.setTextSize(1);
  display.setCursor(0,0);
  display.println("attention !!!"); //выводит на экран надпись
  display.println("heating up");    //внимание !!! идет нагрев

  if (temp > temperature) kettle = false;
}

if (kettle == false) {
  analogWrite(moto, 0);      //отключаем мотор
  digitalWrite(rele, LOW);   //отключаем нагрев
  if (temp < temperature-1) kettle = true;
  };

      time.gettime();

      display.setTextSize(1);
      display.setCursor(5,21);
      display.println("o");
      display.setCursor(0,25);
      display.println("t -");
      display.setCursor(22,25);
      display.println(temp);
      display.println(timer - (time.Hours * 60 + time.minutes));
      display.display();

} while ((time.Hours * 60 + time.minutes) < timer); //если время вышло – завершаем

//делаем дополнительную перестраховку на отключение нагрева и мотора

analogWrite(moto, 0);        //отключаем мотор
digitalWrite(rele, LOW);     //отключаем реле

display.setTextSize(4);
display.setCursor(0,10);
```

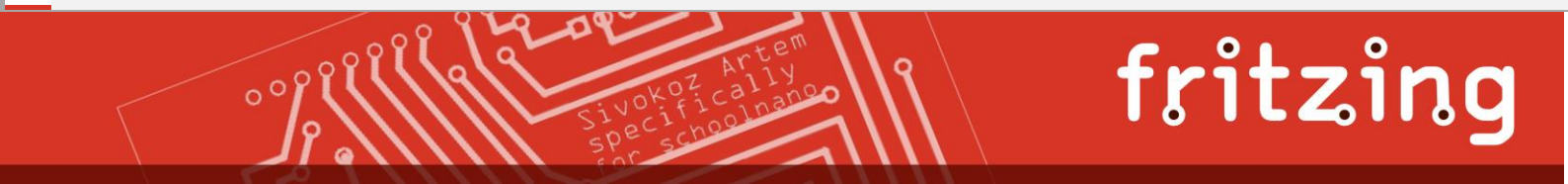


```
271:  display.clearDisplay();
272:  display.println("END");
273:  display.setTextSize(2);
274:  display.setCursor(0,30);
275:  display.println(time.gettime("H:i"));
276:  display.display();
277:  delay(1000000000000000000); //большая задержка что бы приостановить работу программы
278:  }
```

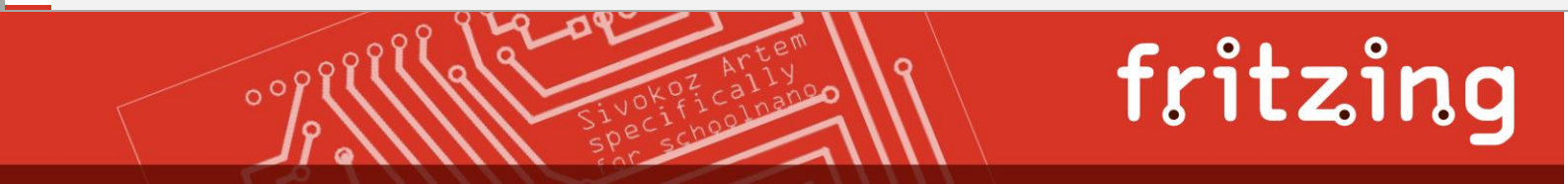
Пробное тестирование выявило не правильную работу нагрева. Оказалось, что после того как достигалась нужная температура и ардуина отключала нагрев, горячая спираль продолжала греть воду. При этом температура воды после отключения могла подыматься на несколько градусов. Отключать нагрев заранее тоже не получилось, потому как при первом, длительном нагреве температура могла уходить до 5 градусов вверх, а при кратковременных нагревах всего на 2-3 градуса.

Помогло изменение алгоритма работы нагрева. Включаем нагрев примерно на одну секунду, после выключаем и ждем около 8 секунд, при этом работает электродвигатель и перемешивает воду. Для этого была добавлена новая переменная int heat

```
#include <math.h> // библиотека для выполнения простых математических операций
#include <RTC.h> //универсальная библиотека для часов реального времени DS1302, DS1307, DS3231
#include <Adafruit_GFX.h> //библиотека для графических дисплеев
#include <Adafruit_PCD8544.h> //для NOKIA LCD 5110

#define moto 9
#define rele 2

int x = 0;
int heat = 0; //вкл. выкл. нагрева (типо шим)
int pinA3 = 0;
int temperature = 36;
int timer = 360;
boolean kettle = true;

RTC time;
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

void setup()   {
 Serial.begin(9600);
 pinMode(A3,INPUT);
 pinMode(moto,OUTPUT);
 pinMode(rele,OUTPUT);
 time.begin(RTC_DS3231);

 display.begin();
 display.clearDisplay();
 display.setContrast(55);
 display.setTextColor(BLACK);

do {
        pinA3 = analogRead(A3);
        Serial.println(analogRead(A3));
        if (pinA3 > 700 && pinA3 < 1100)
                {
                 temperature ++;
                  x=0;
                  delay(200);
                }
        if (pinA3 > 450 && pinA3 < 600)
                {
                 temperature --;
                  x=0;
                  delay(200);
                }
        if (pinA3 > 250 && pinA3 < 400)
                {
                 x++;
                }
        display.clearDisplay();
        display.setCursor(0,0);
        display.setTextSize(1);
        display.println("Temperature:");
        display.setCursor(0,20);
        display.setTextSize(3);
        display.println(temperature);
        display.display();
   }  while (x<5);

x=0;
delay(200);

do {
        pinA3 = analogRead(A3);
        Serial.println(analogRead(A3));
        if (pinA3 > 700 && pinA3 < 1100)
                {
                 timer = timer + 5;
                  delay(200);
                  x=0;
                }

        if (pinA3 > 450 && pinA3 < 600)
```



```
                {
                 timer = timer - 5;
                  delay(200);
                  x=0;
                }
        if (pinA3 > 250 && pinA3 < 400)
                {
                 x++;
                }
        display.clearDisplay();
        display.setCursor(0,0);
        display.setTextSize(1);
        display.println("Time:");
        display.setCursor(0,20);
        display.setTextSize(3);
        display.println(timer);
        display.display();
    }  while (x<5);

time.settime(0,0,0,0,0,0,0); //устанавливаем время в таймере
}


void loop() {

do {
display.clearDisplay();
double temp = analogRead(A0);

  temp = log(((10240000/temp) - 10000));
  temp = 1 / (0.002229327435341785 + (0.0002401007279668999 * temp) + (0.000000020839230886815970 * temp * temp * temp));
  temp = temp - 273.15;

//греем чайник
if (kettle == true) {
  analogWrite(moto, 200); //включаем мотор

  //нагреваем спираль короткими импульсами по несколько секунд
  if (heat < 21) {
       digitalWrite(rele, HIGH);
       heat++;
   }

   //отключаем нагрев
   if (heat > 20 && heat <201) {
       digitalWrite(rele, LOW);
       heat++;
   }

   //сброс счетчика
   if (heat > 200) heat=0;
  }
  display.setTextSize(1);
  display.setCursor(0,0);
  display.println("attention !!!"); //выводит на экран надпись
  display.println("heating up");    //внимание !!! идет нагрев
 if (temp > temperature) kettle = false;

if (kettle == false) {
  analogWrite(moto, 0);      //отключаем мотор
  digitalWrite(rele, LOW);   //отключаем нагрев
  if (temp < temperature-1) kettle = true;
  };

      time.gettime();

      display.setTextSize(1);
      display.setCursor(5,21);
      display.println("o");

      display.setCursor(0,25);
      display.println("t -");
```

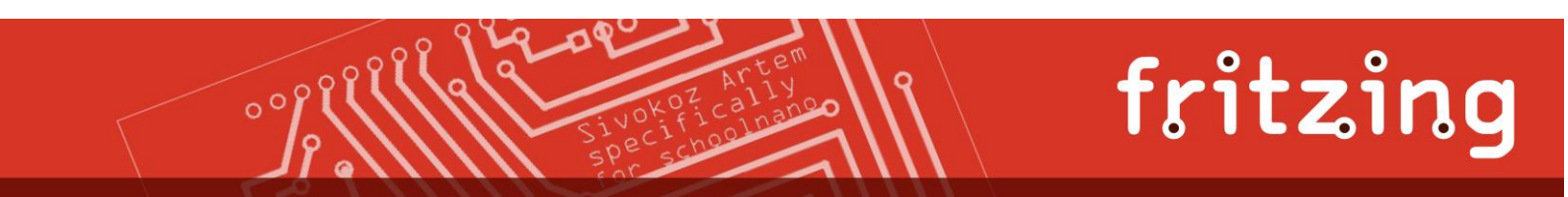


```
424:        display.setCursor(22,25);
425:        display.println(temp);
426:        display.println(timer - (time.Hours * 60 + time.minutes));
427:        display.display();
428:
429:  } while ((time.Hours * 60 + time.minutes) < timer); //если время вышло – завершаем
430:
431:  //делаем дополнительную перестраховку на отключение нагрева и мотора

432:  analogWrite(moto, 0);        //отключаем мотор
433:  digitalWrite(rele, LOW);     //отключаем реле
434:
435:  display.setTextSize(4);
436:  display.setCursor(0,10);
437:  display.clearDisplay();
438:  display.println("END");
439:  display.setTextSize(2);
440:  display.setCursor(0,30);
441:  display.println(time.gettime("H:i"));
442:  display.display();
443:  delay(1000000000000000000); //большая задержка что бы приостановить работу программы
444:  }
```

Таким образом удалось достичь очень плавного разогрева, без превышения установленной температуры.

Но возникла новая проблема.

Теперь прогрев холодной воды в самом начале работы йогуртницы, стал очень длинным. Нужная температура набиралась только в течении 10 минут. Программу пришлось еще не много доработать, сделав непрерывный нагрев чайника до температуры, меньше установленной на 5 градусов.



```
#include <math.h> // библиотека для выполнения простых математических операций
#include <RTC.h> //универсальная библиотека для часов реального времени DS1302, DS1307, DS3231
#include <Adafruit_GFX.h> //библиотека для графических дисплеев
#include <Adafruit_PCD8544.h> //для NOKIA LCD 5110

#define moto 9
#define rele 2

int x = 0;
int heat = 0; //вкл. выкл. нагрева (типо шим)
int pinA3 = 0;
int temperature = 36;
int timer = 360;
boolean kettle = true;

RTC time;
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

void setup()   {
 Serial.begin(9600);
 pinMode(A3,INPUT);
 pinMode(moto,OUTPUT);
 pinMode(rele,OUTPUT);
 time.begin(RTC_DS3231);

 display.begin();
 display.clearDisplay();
 display.setContrast(55);
 display.setTextColor(BLACK);

do {
        pinA3 = analogRead(A3);
        Serial.println(analogRead(A3));
        if (pinA3 > 700 && pinA3 < 1100)
                {
                 temperature ++;
                  x=0;
                  delay(200);
                }
        if (pinA3 > 450 && pinA3 < 600)
                {
                 temperature --;
                  x=0;
                  delay(200);
                }
        if (pinA3 > 250 && pinA3 < 400)
                {
                 x++;
                }
        display.clearDisplay();
        display.setCursor(0,0);
        display.setTextSize(1);
        display.println("Temperature:");
        display.setCursor(0,20);
        display.setTextSize(3);
        display.println(temperature);
        display.display();
   }  while (x<5);

x=0;
delay(200);

do {
        pinA3 = analogRead(A3);
        Serial.println(analogRead(A3));
        if (pinA3 > 700 && pinA3 < 1100)
                {
                 timer = timer + 5;
                  delay(200);
                  x=0;
                }

        if (pinA3 > 450 && pinA3 < 600)
```

```
                {
                 timer = timer - 5;
                  delay(200);
                  x=0;
                }
        if (pinA3 > 250 && pinA3 < 400)
                {
                 x++;
                }
        display.clearDisplay();
        display.setCursor(0,0);
        display.setTextSize(1);
        display.println("Time:");
        display.setCursor(0,20);
        display.setTextSize(3);
        display.println(timer);
        display.display();
    }  while (x<5);

time.settime(0,0,0,0,0,0,0); //устанавливаем время в таймере
}


void loop() {

do {
display.clearDisplay();
double temp = analogRead(A0);

  temp = log(((10240000/temp) - 10000));
  temp = 1 / (0.002229327435341785 + (0.0002401007279668999 * temp) + (0.000000020839230886815 97 * temp * temp * temp));
  temp = temp - 273.15;

//греем чайник
if (kettle == true) {
  analogWrite(moto, 200); //включаем мотор

  //проверяем разницу температур между датчиком и заданной температурой
  //если разница превышает 5 градусов, включаем нагрев на постоянннную работу
  //если разница менее 5 гр. нагреб будет идти короткими импульсами
  if (temp < temperature - 5) {
        digitalWrite(rele, HIGH);
  }
  else {
        //нагреваем спираль короткими импульсами по несколько секунд
        if (heat < 21) {
            digitalWrite(rele, HIGH);
            heat++;
        }

        //отключаем нагрев
        if (heat > 20 && heat <201) {
            digitalWrite(rele, LOW);
            heat++;
        }

        //сброс счетчика
        if (heat > 200) heat=0;
  }
  display.setTextSize(1);
  display.setCursor(0,0);
  display.println("attention !!!"); //выводит на экран надпись
  display.println("heating up");    //внимание !!! идет нагрев
 if (temp > temperature) kettle = false;
}

if (kettle == false) {
  analogWrite(moto, 0);      //отключаем мотор
  digitalWrite(rele, LOW);   //отключаем нагрев

  display.setCursor(0,0);
  if (temp < temperature-1) kettle = true;
```

```
590:    };
591:
592:        time.gettime();
593:
594:        display.setTextSize(1);
595:        display.setCursor(5,21);
596:        display.println("o");
597:        display.setCursor(0,25);
598:        display.println("t -");
599:        display.setCursor(22,25);
600:        display.println(temp);
601:        display.println(timer - (time.Hours * 60 + time.minutes));
602:        display.display();
603:
604:  } while ((time.Hours * 60 + time.minutes) < timer); //если время вышло – завершаем
605:
606:  //делаем дополнительную перестраховку на отключение нагрева и мотора

607:  analogWrite(moto, 0);        //отключаем мотор
608:  digitalWrite(rele, LOW);     //отключаем реле
609:
610:  display.setTextSize(4);
611:  display.setCursor(0,10);
612:  display.clearDisplay();
613:  display.println("END");
614:  display.setTextSize(2);
615:  display.setCursor(0,30);
616:  display.println(time.gettime("H:i"));
617:  display.display();
618:  delay(1000000000000000000); //большая задержка что бы приостановить работу программы
619:  }
```

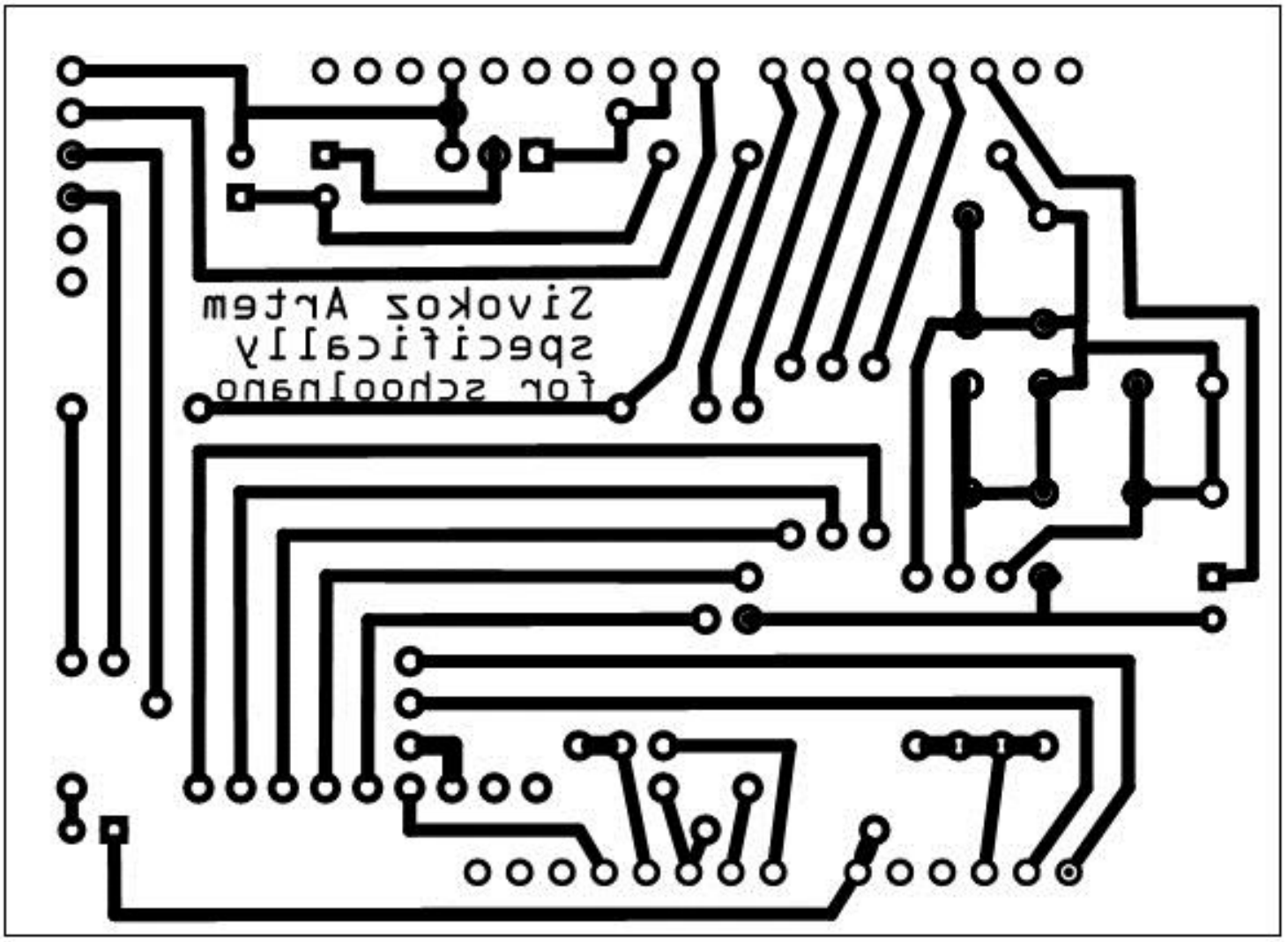
Таким образом получился очень быстрый и точный разогрев.

Теперь, когда йогуртница протестирована и все работает так как нам надо, я приступил к проектированию печатной платы.

Это первая моя плата, выполненная в программе Fritzing. Результат вышел не сразу, пришлось очень много повозиться. После распечатки пробного образца на бумаге, выяснилось, что кнопки хоть и соответствую размеру, но используются другие контакты, а программа категорически не хотела изменять их положение (хотя возможно просто я не до конца в ней разобрался).

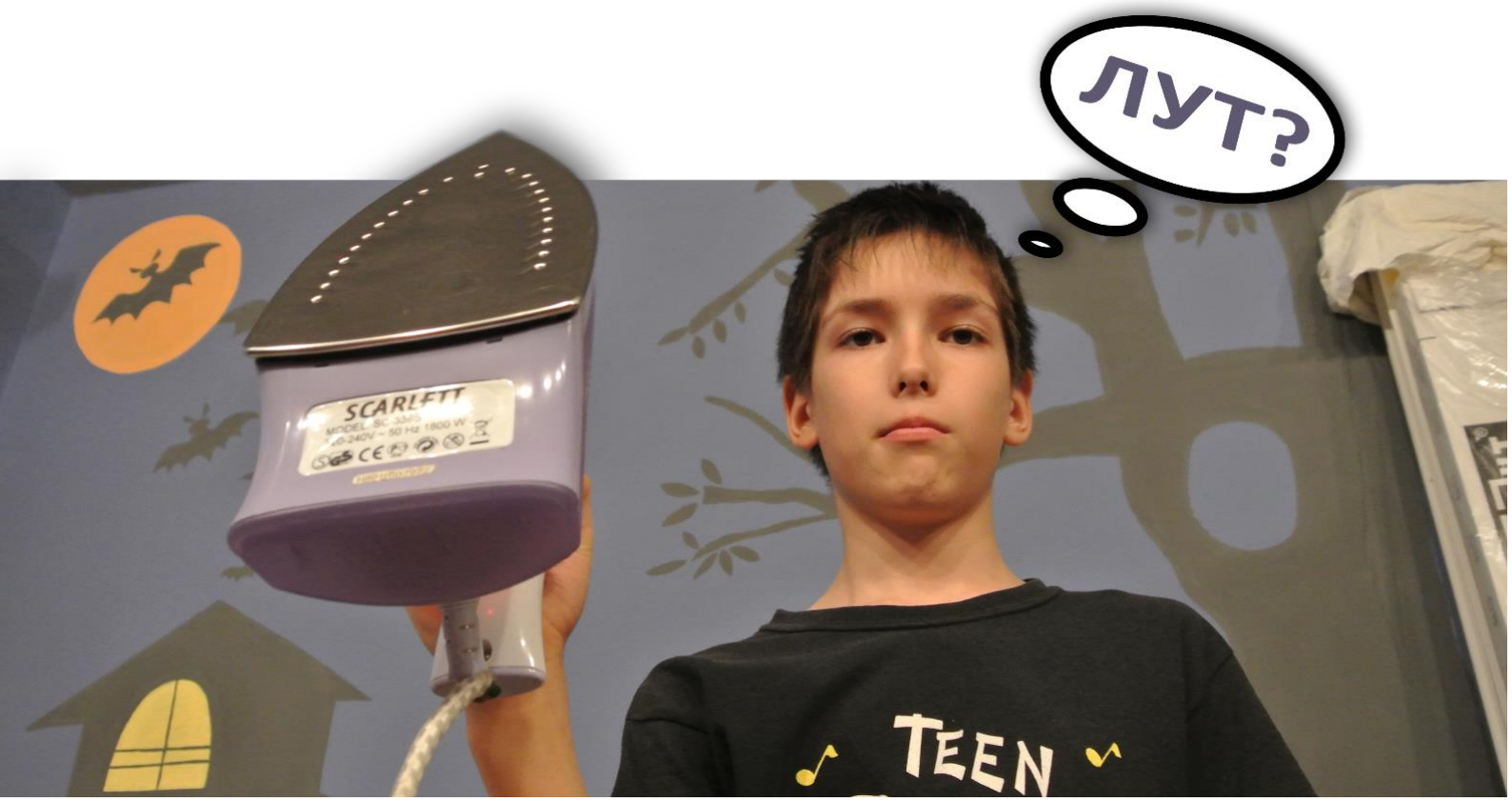
Выходом стало объединение всех дорожек кнопки в одну, для дальнейшего их разрезания под нужную кнопку в момент пайки, на готовой плате.

Вот так выглядит законченный вариант печатной платы!



Получилось на столько красиво, что захотелось воплотить ее в жизнь.

В интернете был найден способ изготовления плат в домашних условиях под названием ЛУТ, что расшифровывается как «Лазерно Утюжная Технология».

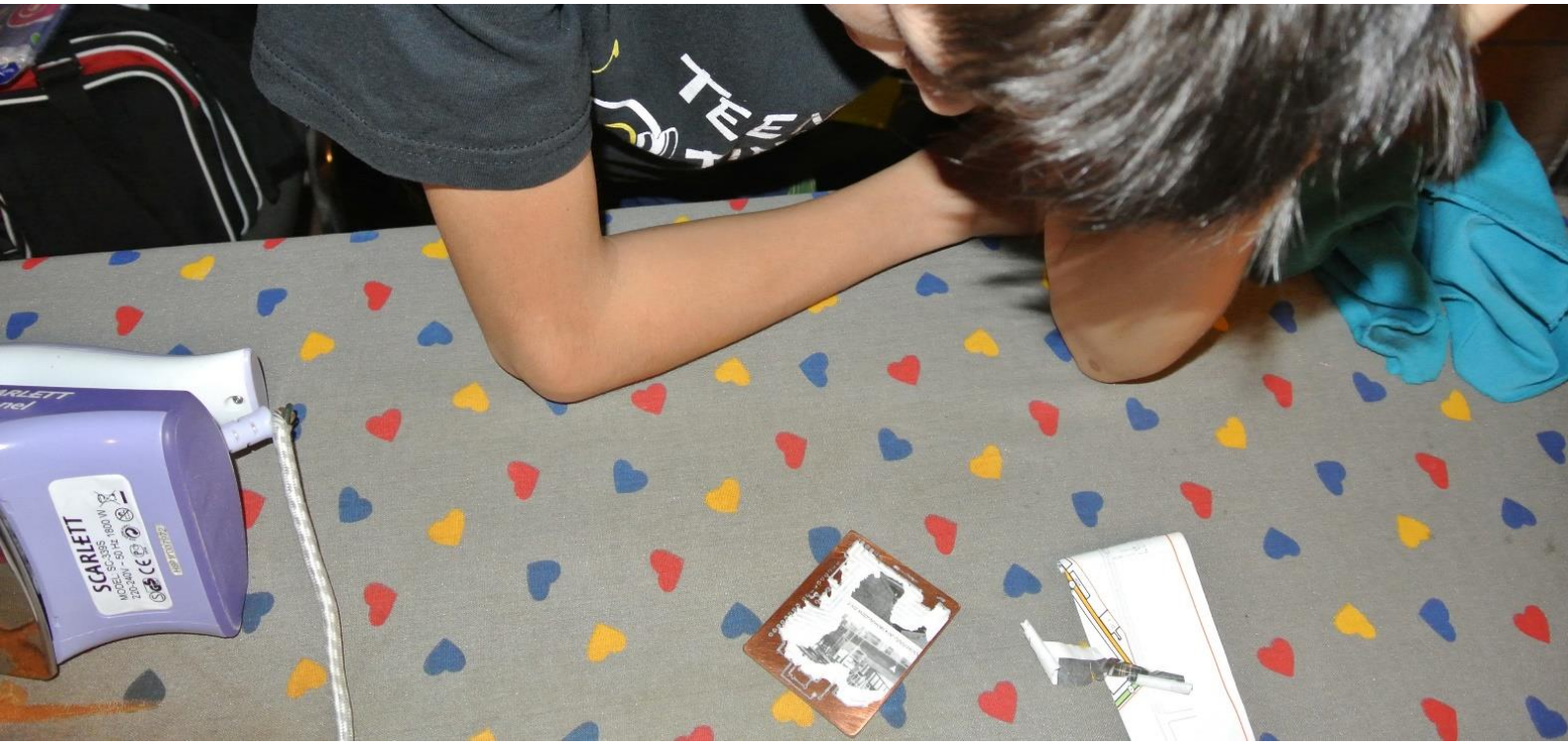


Суть ее состоит в переносе изображения дорожек, напечатанных лазерным принтером с бумаги на плату с последующим вытравливанием в хлорном железе.

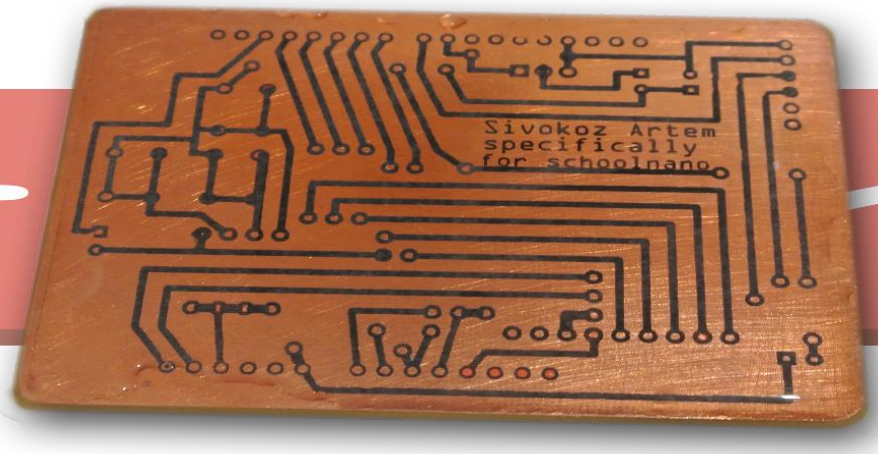
Плата была вырезана по нужному размеру, тщательно обработана наждачной бумагой до зеркального состояния.

В качестве бумаги для переноса изображения, после многих проб и ошибок, была выбрана глянцевая газета под названием «Вальцовка» на плотной бумаге. Изображение с нее переносилось отлично, если только не попадались напечатанные на газете красные полосы, которые очень плотно припекались вместе с напечатанным изображением к плате.

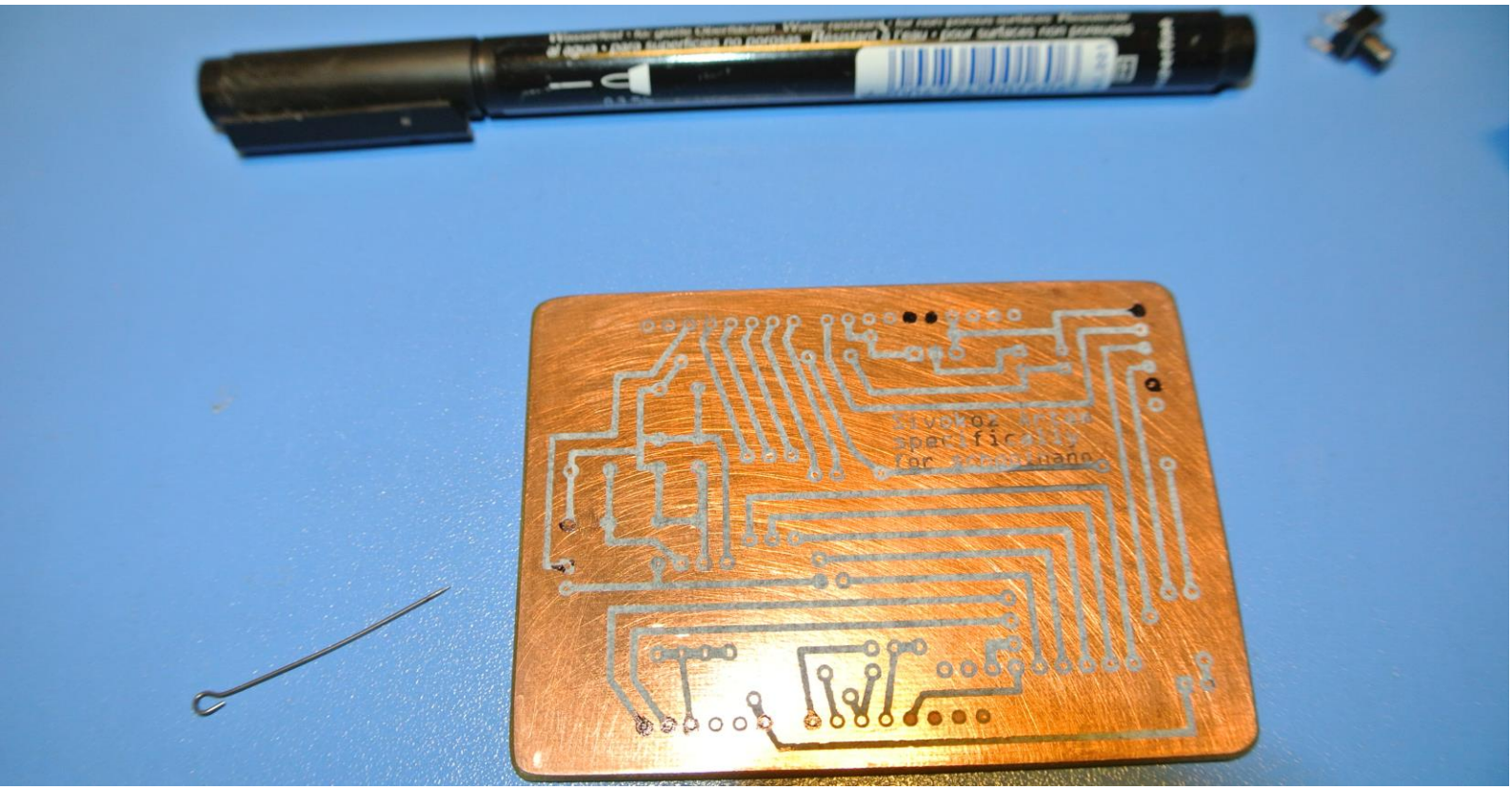
С первого раза ничего не получилось.



Потом был найден свой способ. Оказалось, проще всего предварительно прогреть плату утюгом, следом наложить бумагу с изображением (она сразу же прилипает) быстро пройтись кончиком горячего утюга по всей бумаге и не дожидаясь полного остывания понести плату под холодную воду. Не много подержав в воде, бумага раскисает и хорошо скатывается пальцами, оставляя на плате напечатанные дорожки.



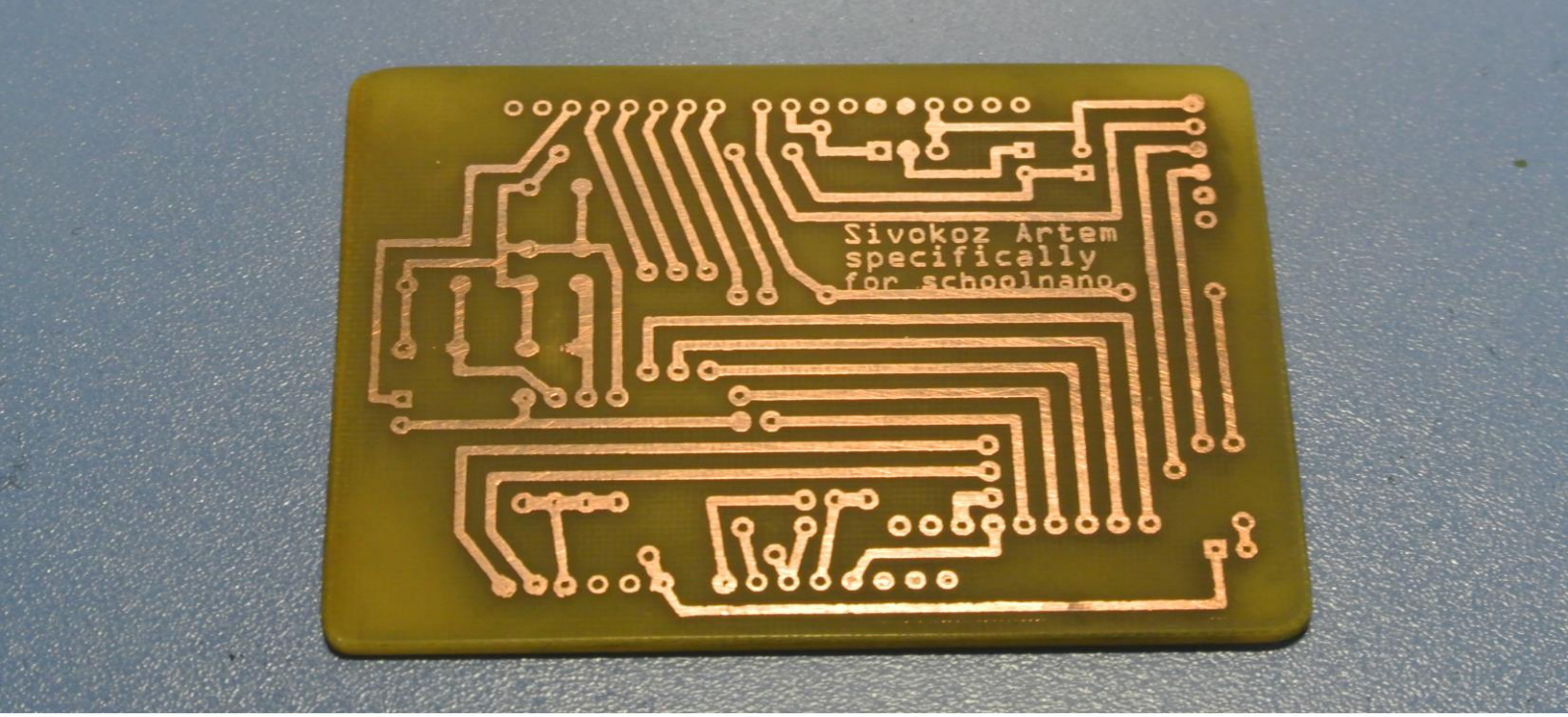
Теперь, перед тем как вытравить плату, необходимо ее немного подправить. Нам понадобится иголка и не стираемый маркер.



Иголкой подтираем лишние дорожки, которые были нарисованы для наших кнопок. А маркером можно немного подрисовать те дорожки, которые плохо пропечатались.

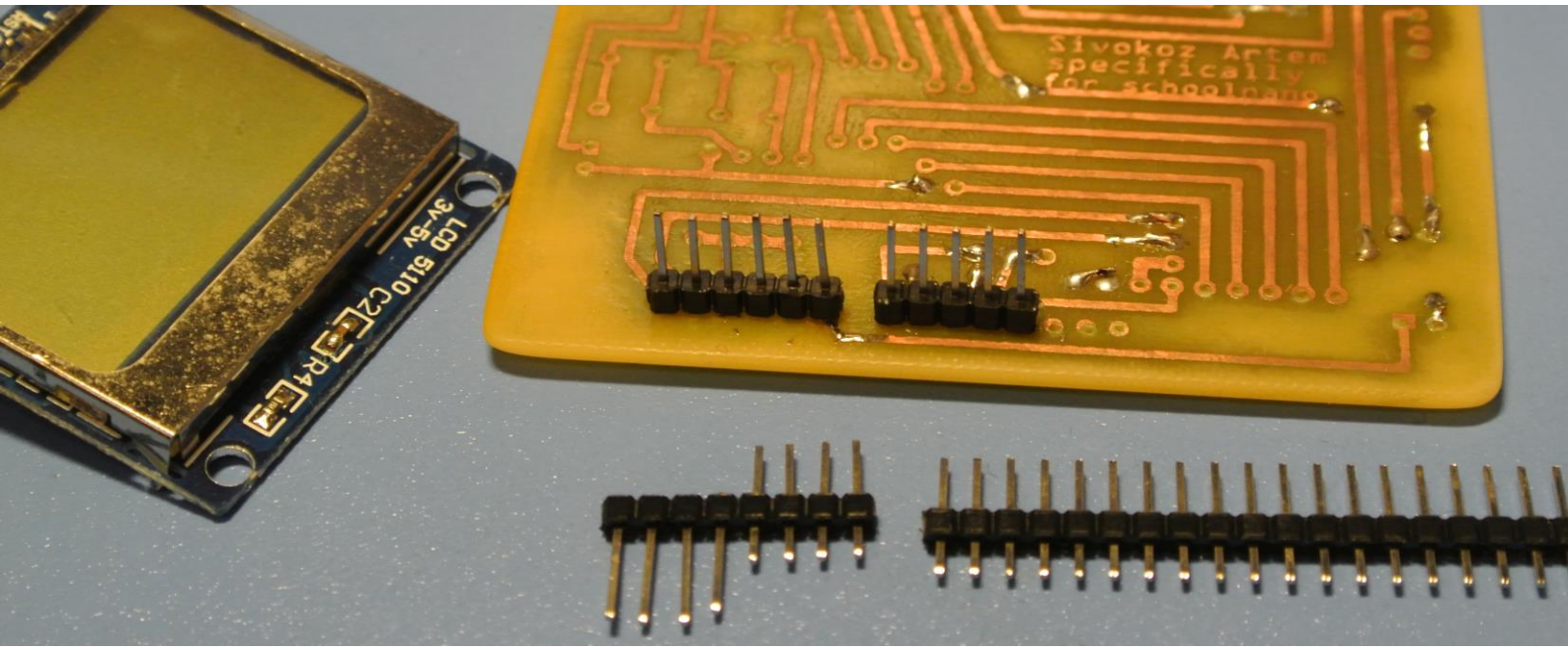
После того как все изменения внесены, опускаем плату в хлорное железо.

Оно разъедает всю медь, за исключением той на которой было нанесено изображение. После вытравливания, получившуюся плату протираем ацетоном смывая тонер принтера и получаем вот такую красоту:

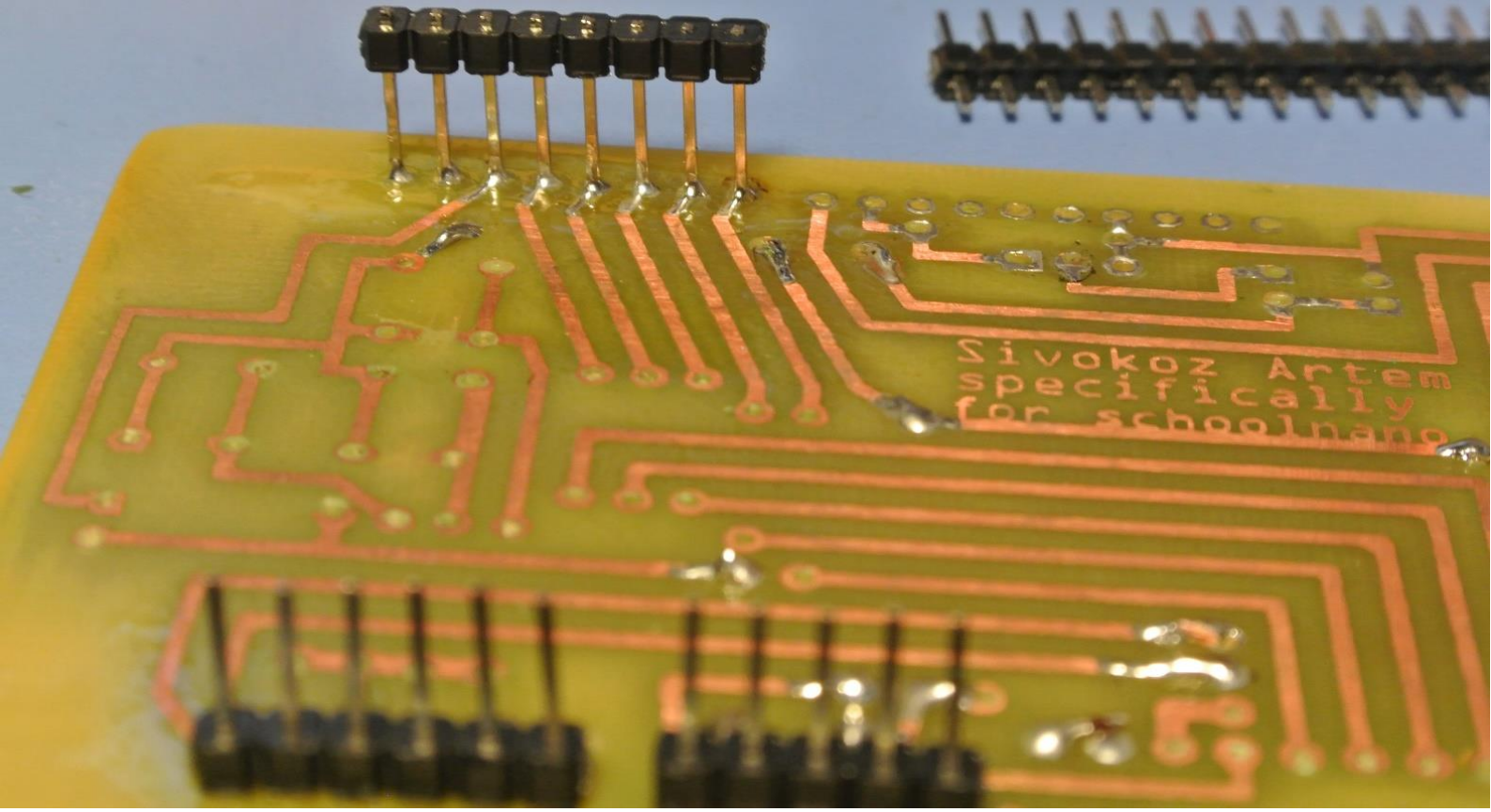


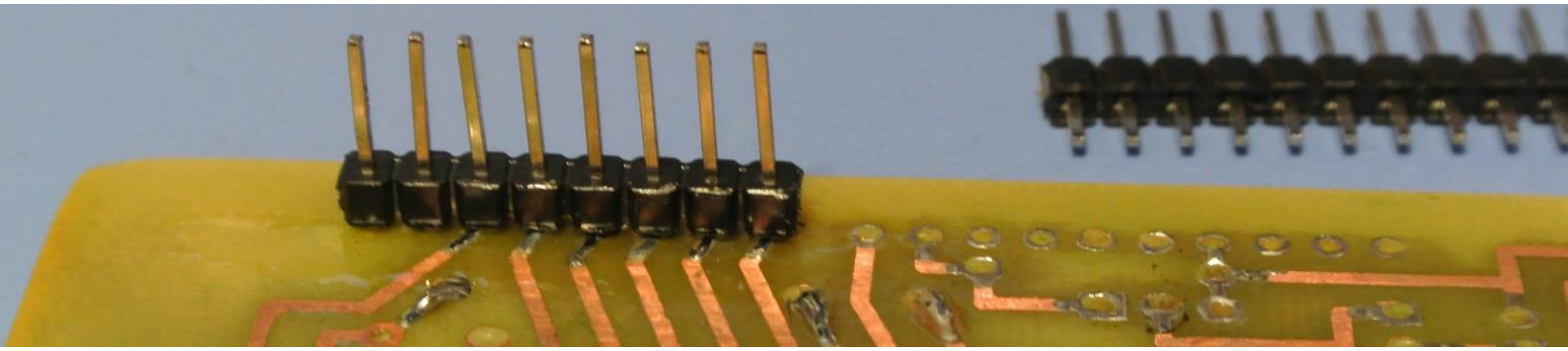
После сверления отверстий приступаем к пайке.

Самым сложным оказалось впаять разъемы, т.к. плата односторонняя, а впаивать пришлось с обратной стороны.

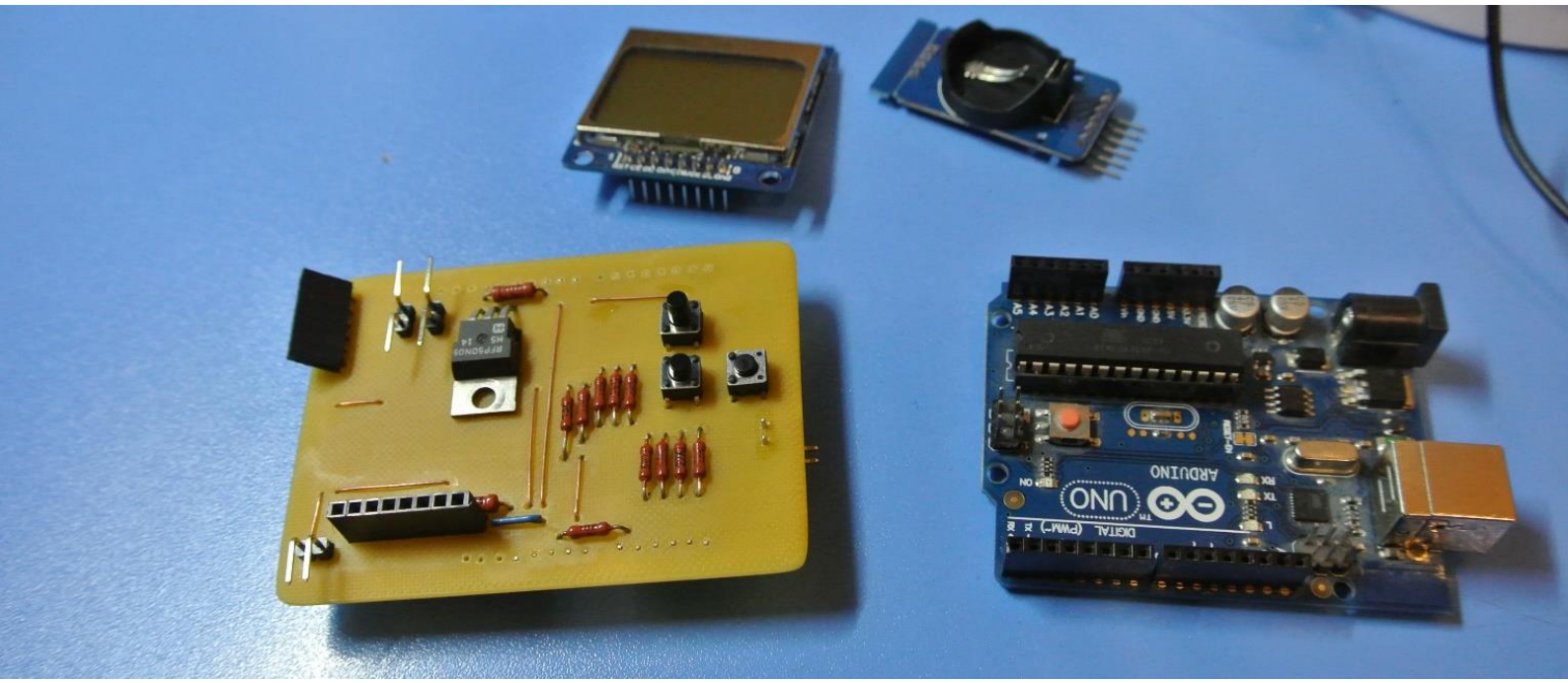


Было найдено решение, убрать пластиковую обойму, впаять отдельно металлические штырьки и потом одеть обойму обратно.

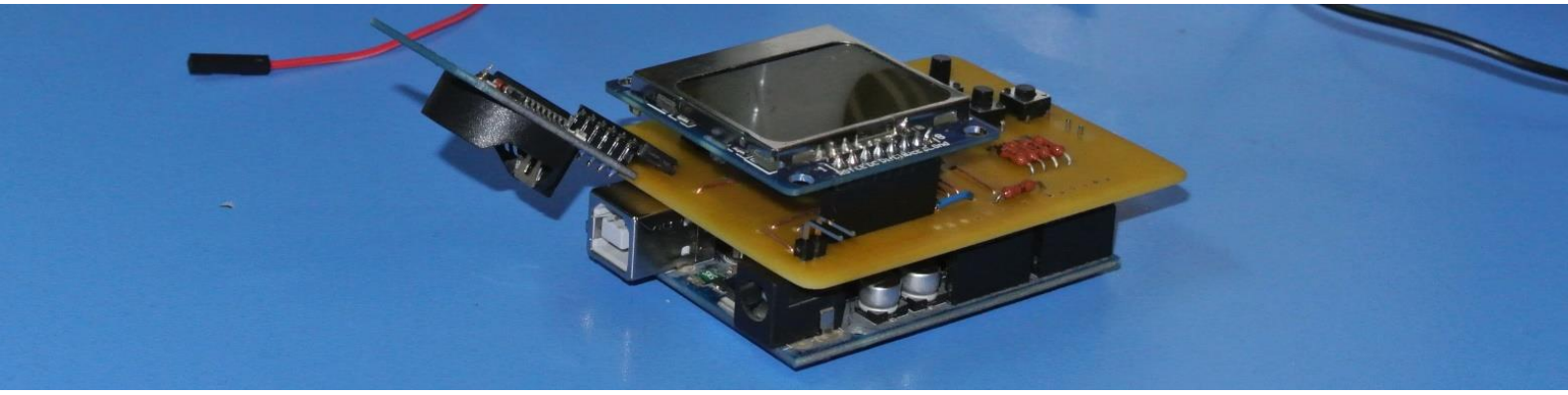


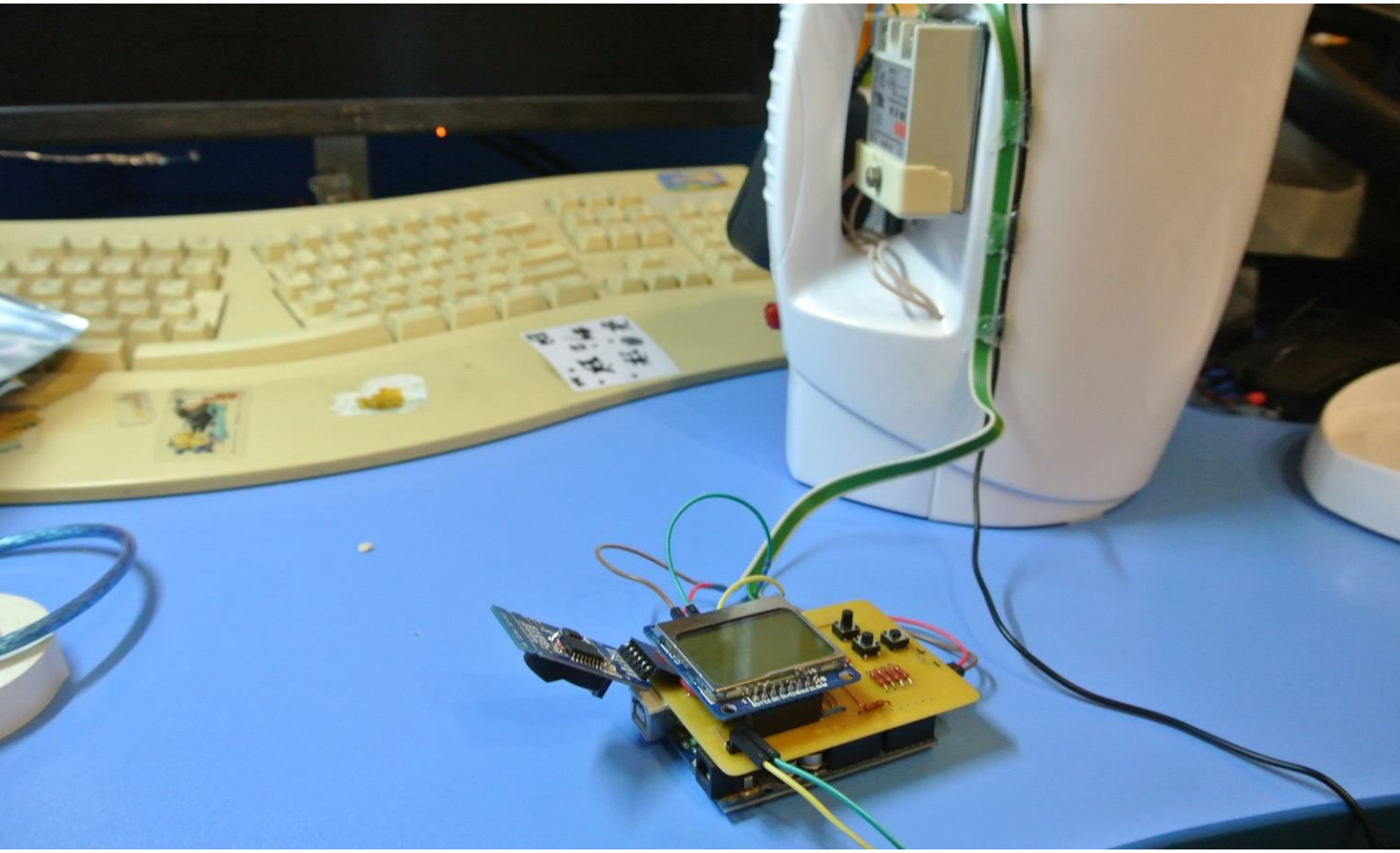


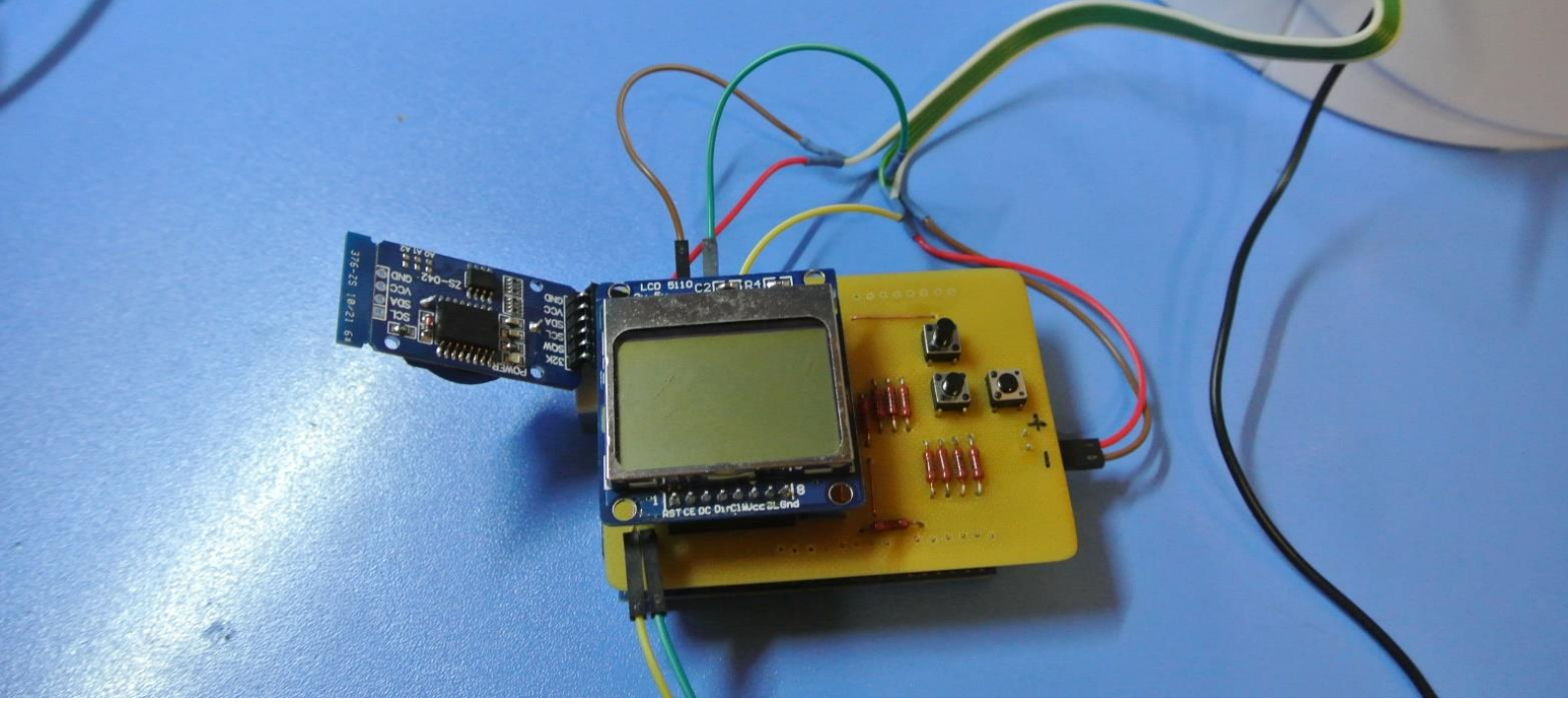
Получилась вот такая красивая плата:



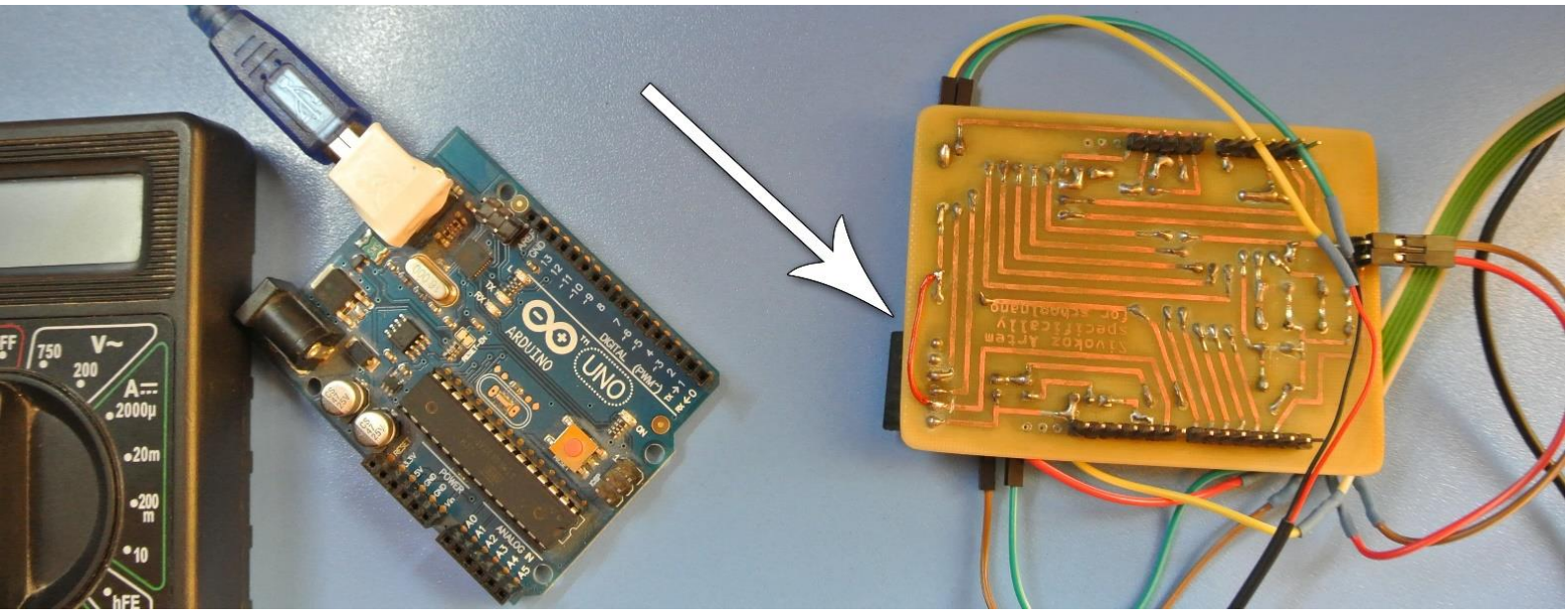
К сожалению, из-за малого опыта в создании подобных плат, не все получилось до конца проработанным. Возникла проблема с размещением таймера реального времени. Он мешал USB разъёму ардуины. Пришлось его отогнуть вверх.







После сборки плата заработала почти правильно, за исключением таймера реального времени. В тестовом варианте питание таймера было подключено напрямую к ардуине, а в процессе создания платы мне захотелось управлять питанием программно и оно было подключено на один из цифровых выходов. Но оказалось, что цифровые выходы сильно слабые и для питания таймера не подходят. Поэтому на плате пришлось перерезать дорожку питания таймера и припаять проводок идущий на 5 вольт.



Отсюда вывод:

Нельзя вносить в проект не опробованные на стадии тестирования изменения

Теперь все готово.

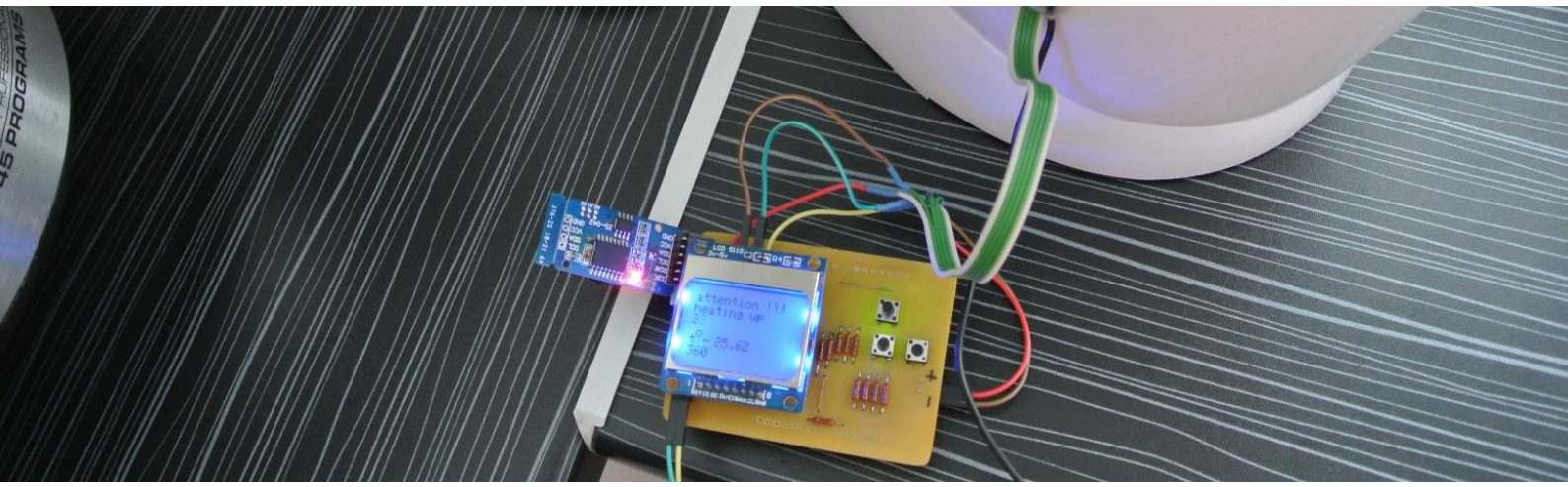
Можно приступать к производству йогурта.

Наполняем чайник водой

Подготавливаем смесь для будущего йогурта состоящую из молока с не большим добавлением натурального йогурта

Погружаем баночку в воду

И включаем йогуртницу



К моменту написания этой статьи в йогуртнице уже было приготовлено больше 10 йогуртов различными способами. Были выявлены и доработаны слабые стороны, такие как, очень медленный первоначальный нагрев, когда температура воды значительно меньше установленной температуры. Программно был реализован непрерывный нагрев до температуры ниже установленной на 5 градусов, после чего уже программа переходила в режим плавного разогрева.

К сожалению, в рамках конкурса не хватило времени на реализацию всего задуманного. Но в дальнейшем планируется добавить в йогуртницу режим пастеризации молока, для возможности использования парного деревенского молока без предварительного кипячения, а также изменить процесс размешивания воды двигателем, сделав циркуляцию воды не только по кругу, но и сверху в низ.

При этом попытка добавить режим пастеризации в проект программным путем уже предпринималась. Программа разогревала воду до 75 – 80 градусов и должна была выдерживать эту температуру в течении получаса. Но применяемый в проекте термоклей не выдержал этой температуры и все стало отклеиваться. Поэтому с режимом пастеризации решено было повременить.

Результат работы йогуртницы:

В конце хочу сказать большое спасибо своему папе за оказанную мне помощь в реализации этого проекта.