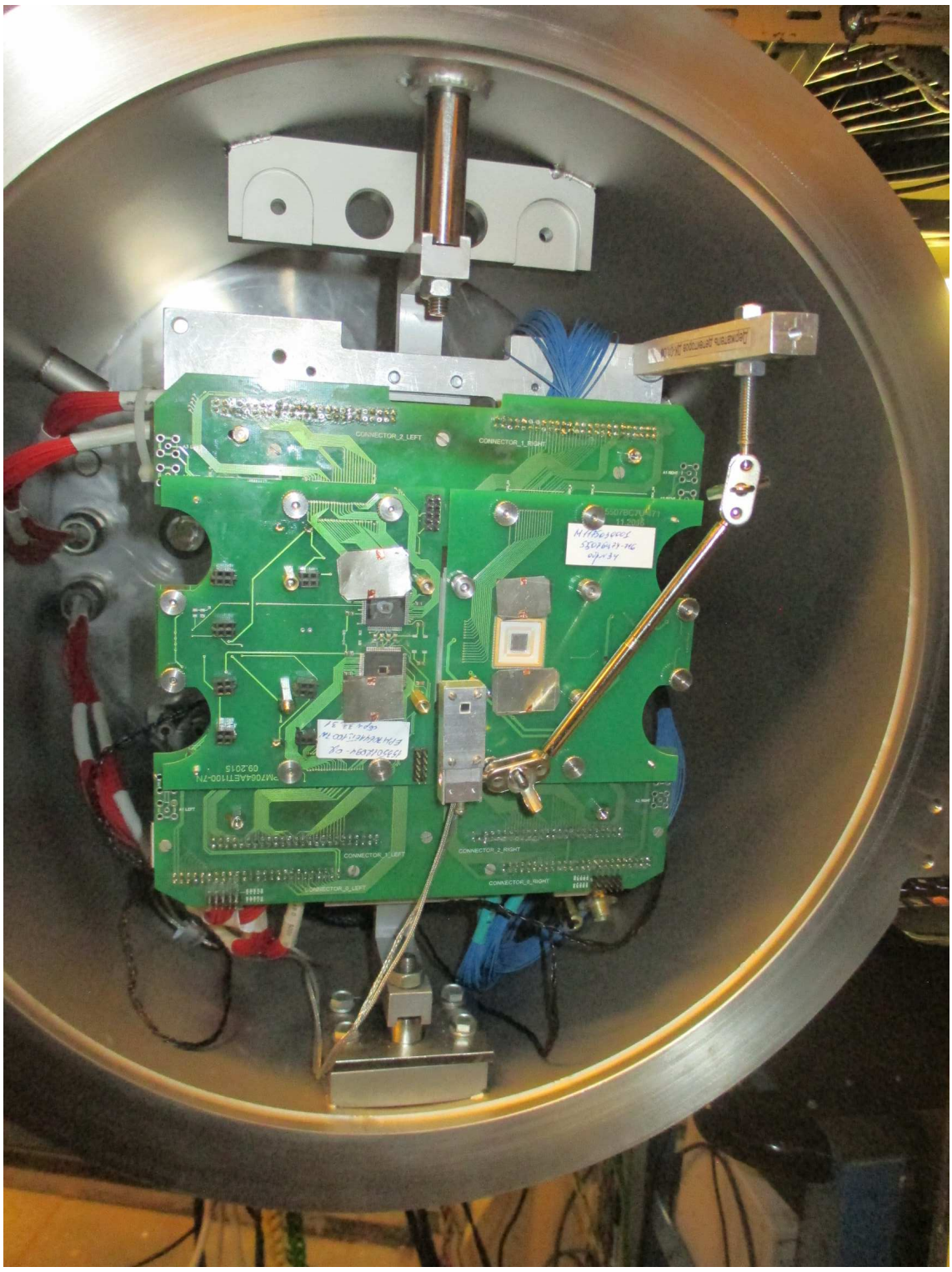


Рисунок 1.1 — Вакуумная камера с установленными микросхемами

Рисунок 1.2 — Корпус, в котором содержится ТРИД и управляющая система

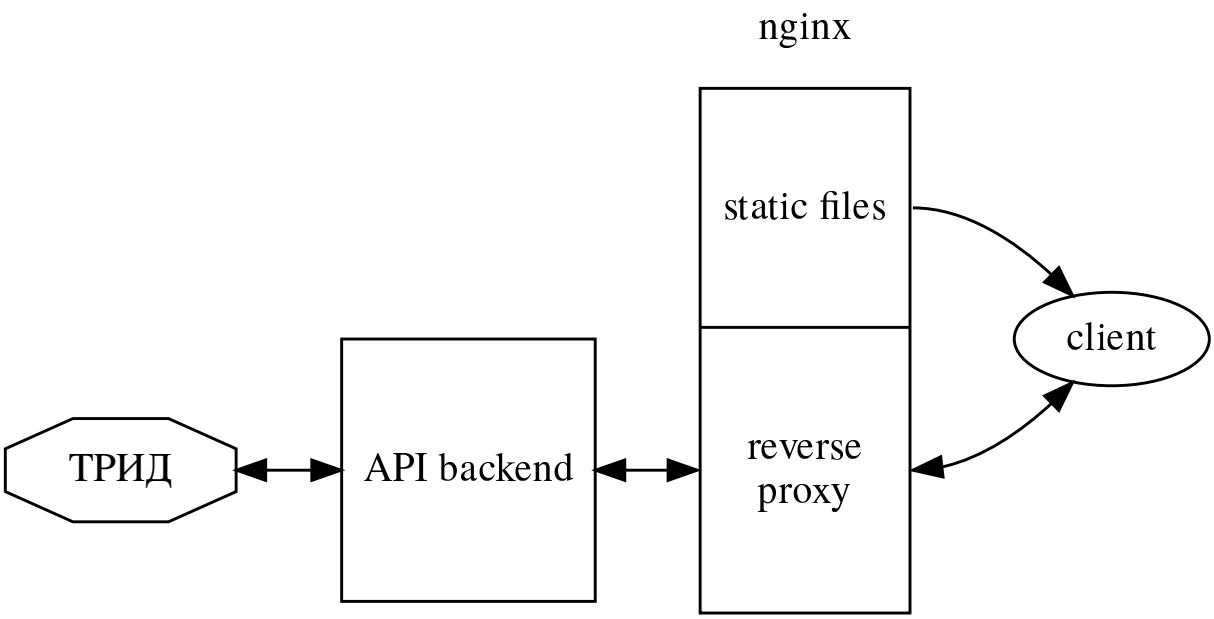


Рисунок 2.1 — Архитектура Web-сервиса

2.4 Выбор библиотек Python

Из предыдущего раздела видно, что для работы API backend требуются следующие библиотеки:

а) Библиотека для работы с последовательным портом ввода-вывода.
б) Библиотека для работы с протоколом modbus (поверх последовательного порта).
в) Библиотека для сбора отладочной информации (создания журнала).
г) Библиотека для создания Web сервиса.

В случае с библиотекой для работы с последовательным портом ввода-вывода есть фактически единственная альтернатива — pyserial[13], других развивающихся проектов данной тематики найти не удалось.

Протокол modbus поддерживается следующими библиотеками: MinimalModbus[14], pylibmodbus[15], modbus_tk[16], pymodbus[17], uModbus[18]. Из них pylibmodbus была отброшена из-за требования наличия дополнительной библиотеки на C, длительного отсутствия обновлений и отсутствия документации. Modbus_tk также отличалась отсутствием документации, а официальный пакет pymodbus в PyPI не поддерживается, хотя сама библиотека развивается[19]. Среди оставшихся MinimalModbus и uModbus первая была выбрана за более удобный интерфейс.

В качестве библиотеки для сбора отладочной информации оказалось возможным использовать часть стандартной библиотеки Python, предназначеную для этой цели. На nginx была дополнительно возложена обязанность предоставления создаваемых данной библиотекой журналов.

3 Технологический раздел

3.1 Архитектура приложения

Как было показано в пункте 2.3, приложение разбито на два основных модуля: nginx и API backend. В данном разделе будет рассмотрена только архитектура API backend, настройка nginx рассматривается отдельно в пункте 3.2 а архитектура этой части тривиальна и фактически отображена на рисунке 2.1.

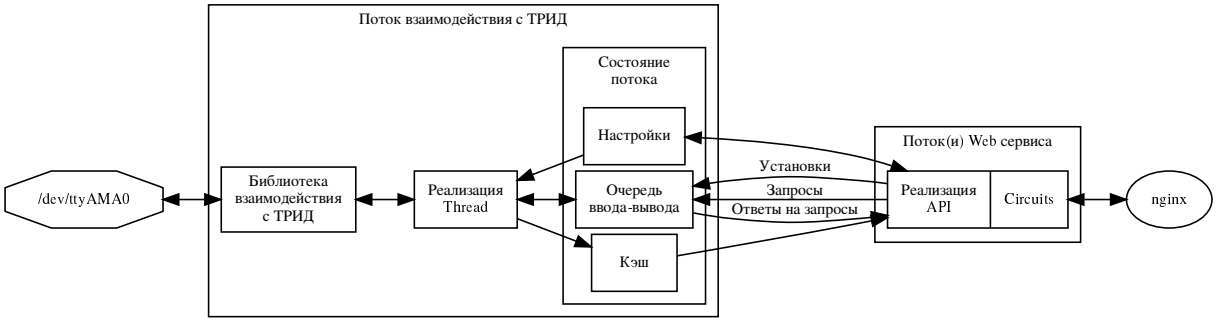


Рисунок 3.1 — Архитектура API backend

Архитектура API backend изображена на рис. 3.1. Всё приложение было разбито на три основных модуля: выделенная библиотека взаимодействия с ТРИД (используется внутри потока ввода-вывода), реализация потока ввода-вывода на основе классов из стандартной библиотеки Python — модуля **threading** и собственно реализации API на основе классов из библиотеки **circuits**.

Такая архитектура была создана из-за наличия только одной дуплексной линии связи с ТРИД: необходимо обеспечить строгую последовательность отправки запросов к ТРИД и получения ответов на случай прихода нового запроса от nginx во время обработки старого. Данную проблему можно решить, как минимум, на трёх уровнях: запретить nginx создавать более одного параллельного соединения к Web серверу, не использовать потоки при обработке запросов или работать с ТРИД исключительно через отдельный поток и очередь и/или блокировки. Третий вариант был выбран, из-за того, что существуют запросы, одновременная обработка которых безопасна. В основном это относится к запросам, получающим значения из кэша или настроек. Помимо этого поток взаимодействия с ТРИД требуется, чтобы постоянно опрашивать ТРИД, узнавая показания температурного датчика; без отдельного потока эту задачу пришлось бы возложить на Web-интерфейс, что привело бы к его замедлению.

3.2 Настройка nginx и circuits

В соответствие с рис. 2.1 nginx должен выполнять две функции: предоставлять доступ к статическим файлам (в них определяется весь Web-интерфейс) и предоставлять

3.7 Web-интерфейс

Описанный выше интерфейс (API) пригоден для использования программами, но он не предназначен для взаимодействия с человеком. В соответствии с пунктом 2.1 для выполнения данной задачи используется nginx, настроенный на раздачу статических файлов, предполагающих использование браузера. Всего для создания Web-интерфейса потребовалось четыре файла:

а) Описание страницы, отображаемой браузером на языке разметки HTML. Размещается в **html/trid.html** и является основной точкой входа.

б) Стилевой файл в формате CSS. Размещается в **html/static/style.css**, управляет отображением страницы.

в) Описание поведения страницы в формате JavaScript. Размещается в **html/static/main.js**.

г) Библиотека dygraph.js (минифицированная и с встроенными в файл зависимостями).

3.7.1 Особенности HTML-кода страницы

Основная и единственная страница Web-интерфейса содержит простейшую иерархию HTML5-тегов, в основном состоящую из тегов **div**, **input** и **form**. Внешний вид страницы показан на рис. 3.2 и 3.3.

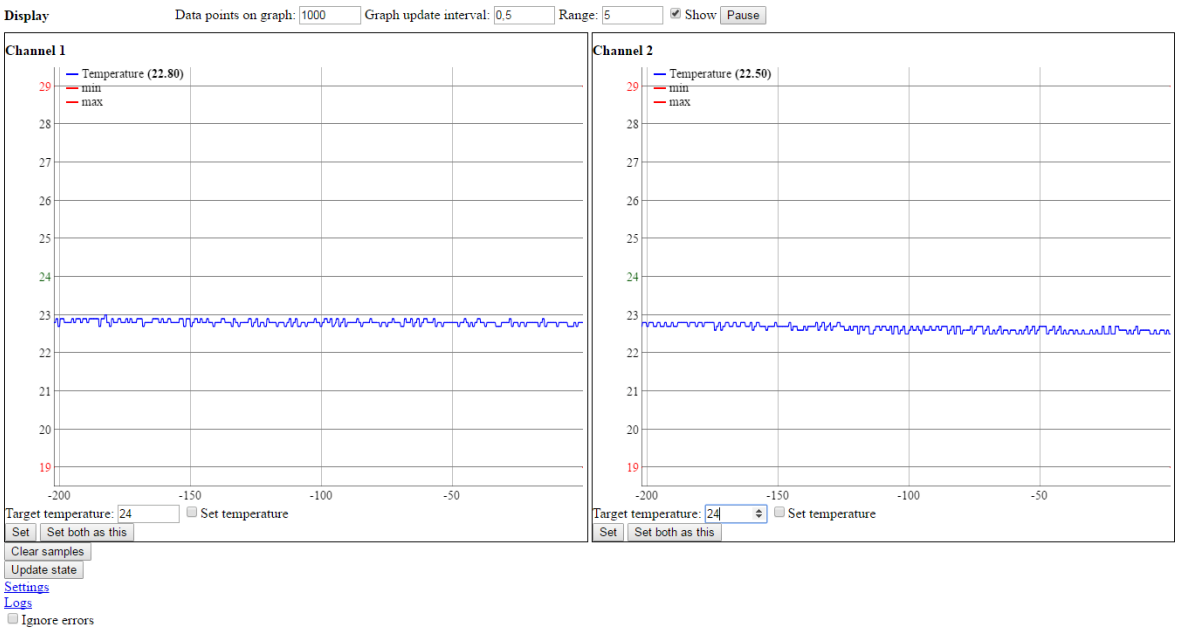


Рисунок 3.2 — Web-интерфейс, основной вид

Как видно из рис. 3.2 и 3.3, страница имеет два основных режима отображения: основной вид, где видны только собственные настройки Web-интерфейса, несколько ссы-

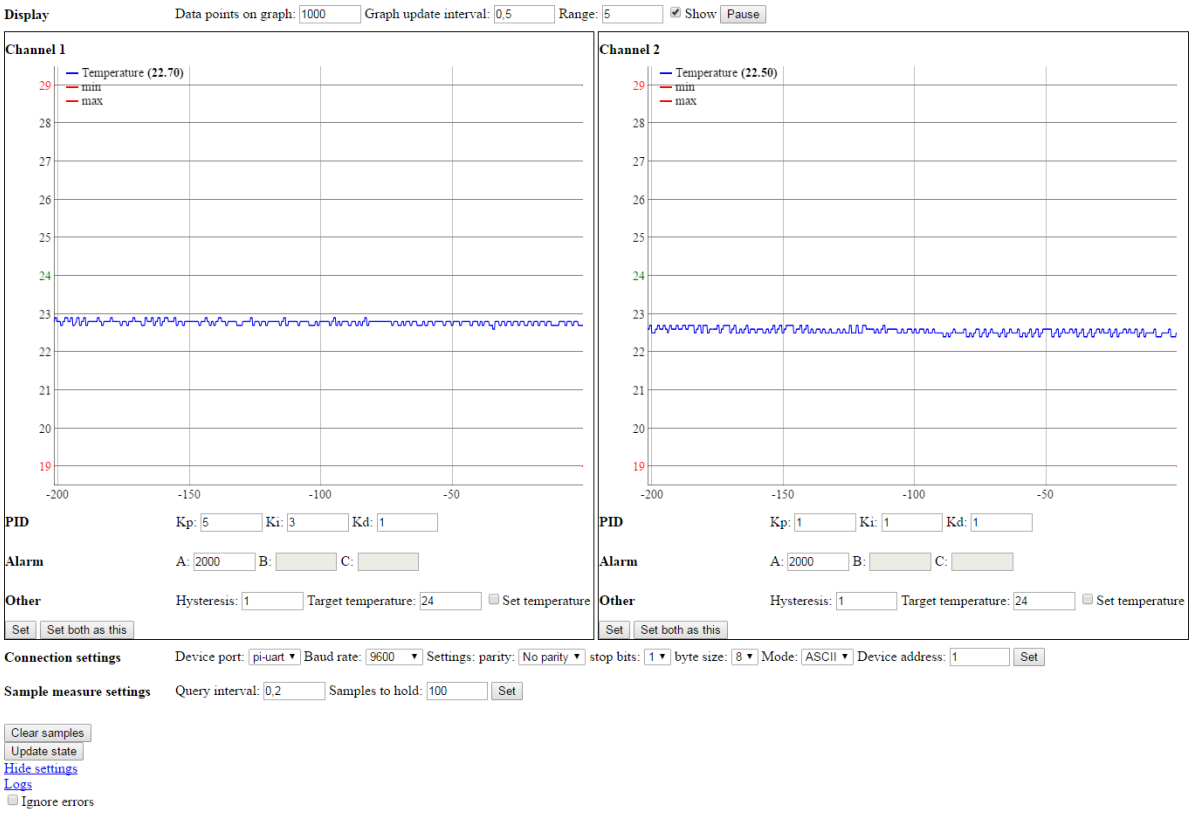


Рисунок 3.3 — Web-интерфейс, вид со всеми доступными настройками

лок на дополнительные режимы и установка целевой температуры, и расширенный вид со всеми параметрами ТРИД, доступными для установки через его RS485 интерфейс. Данное разделение было сделано, чтобы не отвлекать внимание оператора на настройки, изменение которых, скорее всего, не понадобится в ходе эксперимента.

Смена режимов осуществляется путём изменения видимости элементов, а именно манипуляциями CSS атрибутом **display** из JavaScript кода. В связи с тем, что осуществление данных манипуляций напрямую довольно неудобно, JavaScript изменяет атрибут **display** путём присвоения или удаления CSS класса у выбранных тегов **div**.

В HTML коде также определены реакции на пользовательский ввод: а именно

а) Изменения собственных настроек Web-интерфейса применяются немедленно. Дополнительно при нажатии кнопки «ввод» на клавиатуре при поле настройки (если она представляет собой поле для ввода) в фокусе также происходит применение настройки, на случай если событие «изменение поля» не работает. Пример определения поля с собственной настройкой представлен в листинге 14. Пример определения булевой собственной настройки Web-интерфейса представлен в листинге 15.

б) Изменения параметров ТРИД применяются после нажатия кнопки «ввод» на клавиатуре при соответствующем поле в фокусе, либо после использования кнопок «Set» или «Set both as this». Код всех таких параметров для одного канала представлен в листинге 16.

HTML код полей ввода всех описанные выше настроек, кроме «Ignore errors» приведён в листинге 43. Код одного из флажков, соответствующих настройке «Ignore errors» есть в листинге 21.

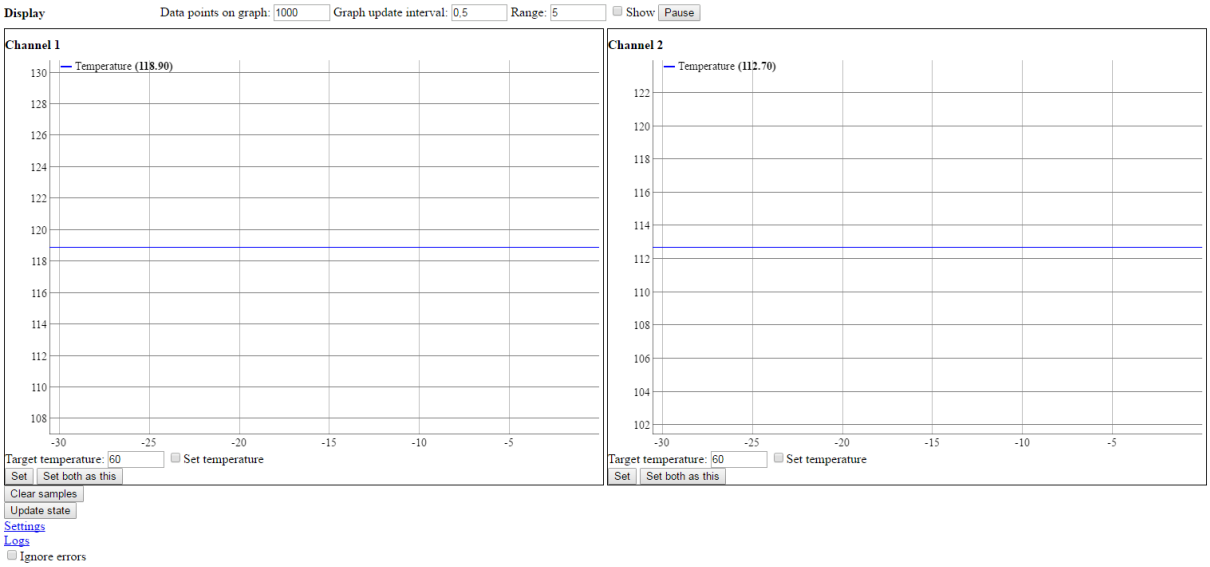


Рисунок 3.4 — Внешний вид Web-интерфейса с отключённым показом диапозона

3.7.3 Описание CSS кода страницы

Использованные CSS классы представлены в листинге 22. Далее приводится их описание:

Класс **control** используется для тегов **form**, позволяя размещать две формы в одной строчке без привлечения дополнительных контейнеров.

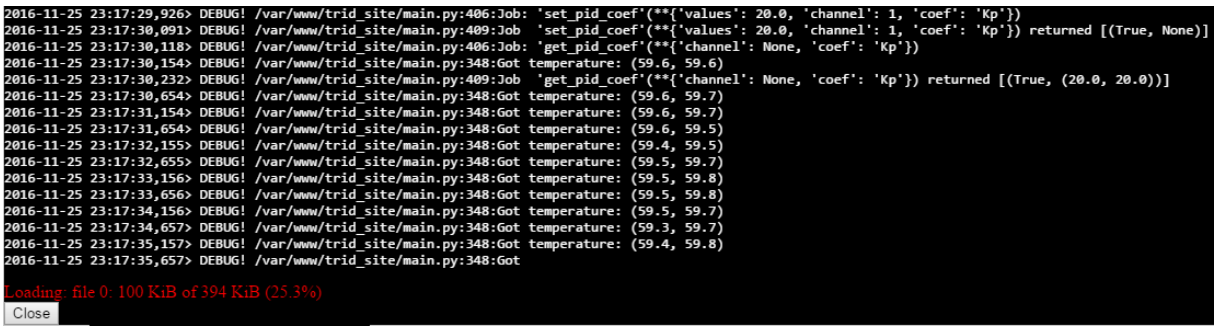
Класс **num-input** управляет отображением полей ввода, используемых для ввода чисел. Благодаря ему все поля ввода чисел имеют одинаковую ширину.

Класс **control-header** используется для жирных подписей около строчек с параметрами или настройками: «Display», «PID», «Alarm», «Other», «Connection settings», «Sample measure settings». Позволяет размещать следующие элементы, с одной стороны, в одной линии с подписью, с другой стороны, точно друг под другом.

Класс **ch-block** применяется для контейнера, содержащего всё, связанное с одним каналом (т.е. график и параметры, а также заголовок вида «Channel 1»). Применяется для рисования единственных двух рамок.

Класс **advanced** применяется для скрытия определённых параметров в обычном режиме и отображения их в режиме с расширенными настройками. Скрытие происходит, если элемент с данным классом попадает внутрь контейнера с классом **advanced-disabler**.

Класс **graph** задаёт размеры графика: 48 % от экрана по ширине, 50 % по высоте. Два процента по ширине зарезервированы под рамки.



а

б

в

Рисунок 3.5 — Режим показа журнала:
а. в процессе загрузки последнего файла: последняя строка журнала ещё не загрузилась полностью, показан индикатор
б. в процессе загрузки предпоследнего файла
в. после загрузки последнего файла

3.7.4.6 Функции режима показа ошибки

Для показа ошибки используется всего одна функция **showError**, задача которой состоит в следующем:

а) Сохранить ошибку в JavaScript консоли вызовом метода браузера **console.log**.

б) Сохранить ошибку в контейнере с идентификатором (атрибутом **id**) **error-message** для показа пользователю.

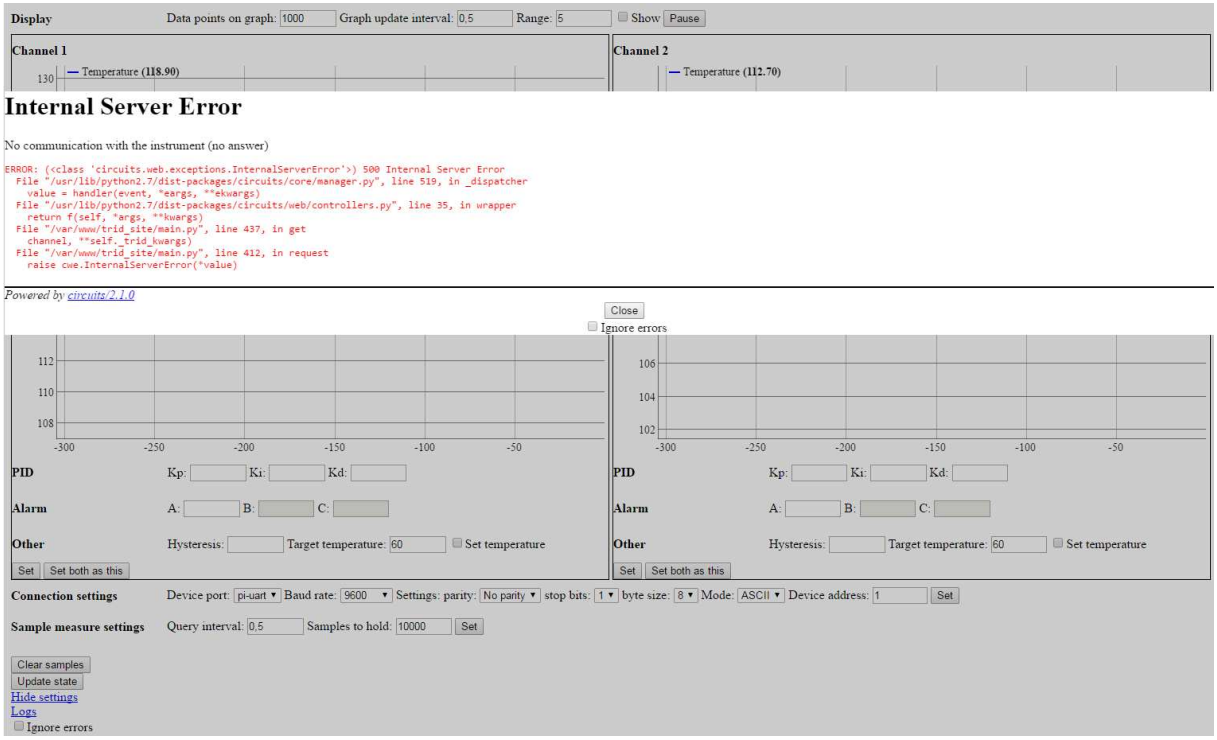


Рисунок 3.6 — Режим показа ошибки: ошибка, отображаемая при разрыве связи с ТРИД

в) Если переменная состояния **ignoreErrors** ложна (т.е. настройка «Ignore errors» не установлена, см. пункт 3.7.4.2), то включить режим отображения ошибки путём присвоения класса **error** контейнеру с идентификатором **error**.

Код данной функции представлен в листинге 35.

```
var eDiv = document.getElementById('error');

var showError = function(html, log) {
    console.log.apply(this, log);
    var eMsgDiv = document.getElementById('error-message');
    eMsgDiv.innerHTML = html;
    if (!ignoreErrors) {
        eDiv.classList = ['error'];
    }
};
```

Листинг 35 — Код функции **showError**

Закрытие режима отображения ошибки осуществляется просто восстановлением класса **error-disabled** у контейнера с идентификатором **error**. Код соответствующей фунции представлен в листинге 54.

Пример отображаемой ошибки показан на рис. 3.6.

ПИД-регулятор — это устройство в управляющем контуре с обратной связью, используемое в системе автоматического управления для формирования управляющего сигнала[25]. Назначение регулятора в данном случае — поддержание заданного значения температуры ИС путём её подогрева с заданной регулятором интенсивностью, интенсивность нагрева задётся методом ШИМ.

Управляющий сигнал (скважность импульсов) при использовании ПИД-регулирования задётся как

$$S = \min\left\{\max\left\{\left(K'_p \Delta T(t) + K'_i \int_{t_0}^{t} \Delta T(\tau) d\tau + K'_d \frac{d\Delta T}{dt}\right); 0\right\}; 1\right\} \tag{4.1}$$

где S — скважность, может находится только в отрезке $[1, 0]$; K'_p, K'_i и K'_d — числа, соответствующие введённым оператором коэффициентам ПИД-регулирования; $\Delta T(t)$ — функция разницы между целевым значением температуры, введённым оператором, и измеряемым значением; а t_0 — некоторый момент во времени, используемый ПИД-регулятором для начала отсчёта. К сожалению, инструкция ТРИД не содержит подробной формулы, позволяющей определить способ интегрирования, момент t_0 или связь между введёнными оператором значениями ПИД-коэффициентов и числами K'_p, K'_i и K'_d, поэтому настройку ТРИД приходится осуществлять экспериментальным путём.

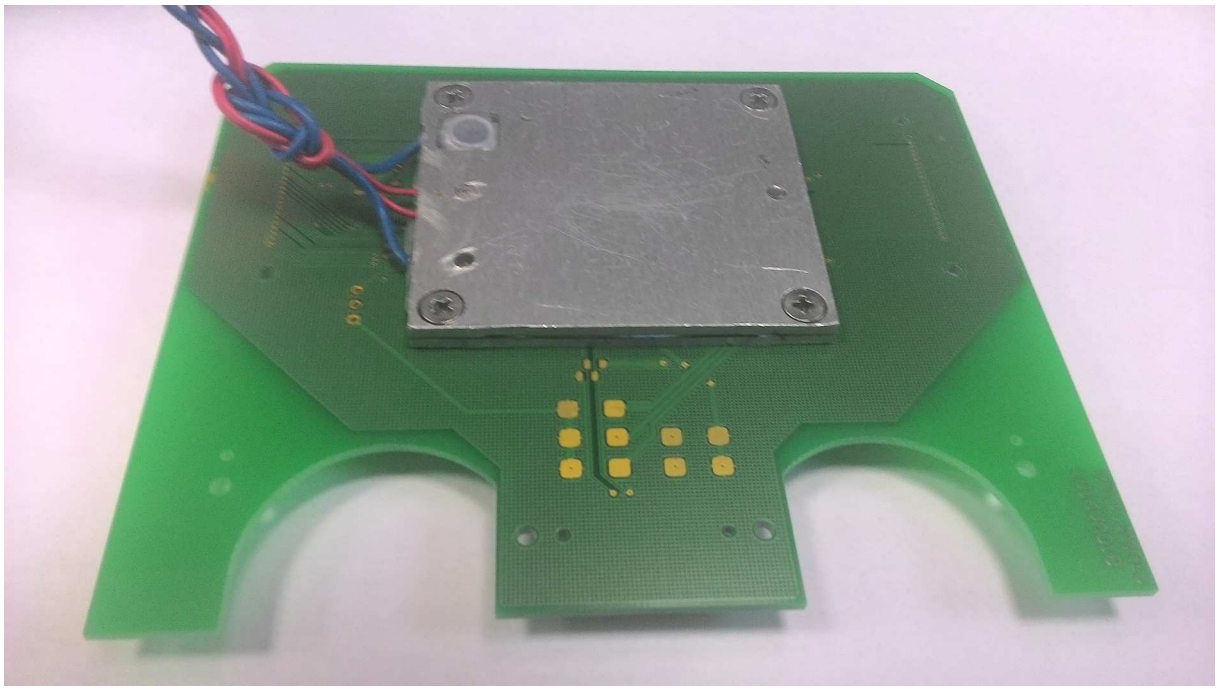


Рисунок 4.1 — Нераспаянная печатная плата с присоединённым нагревательным элементом

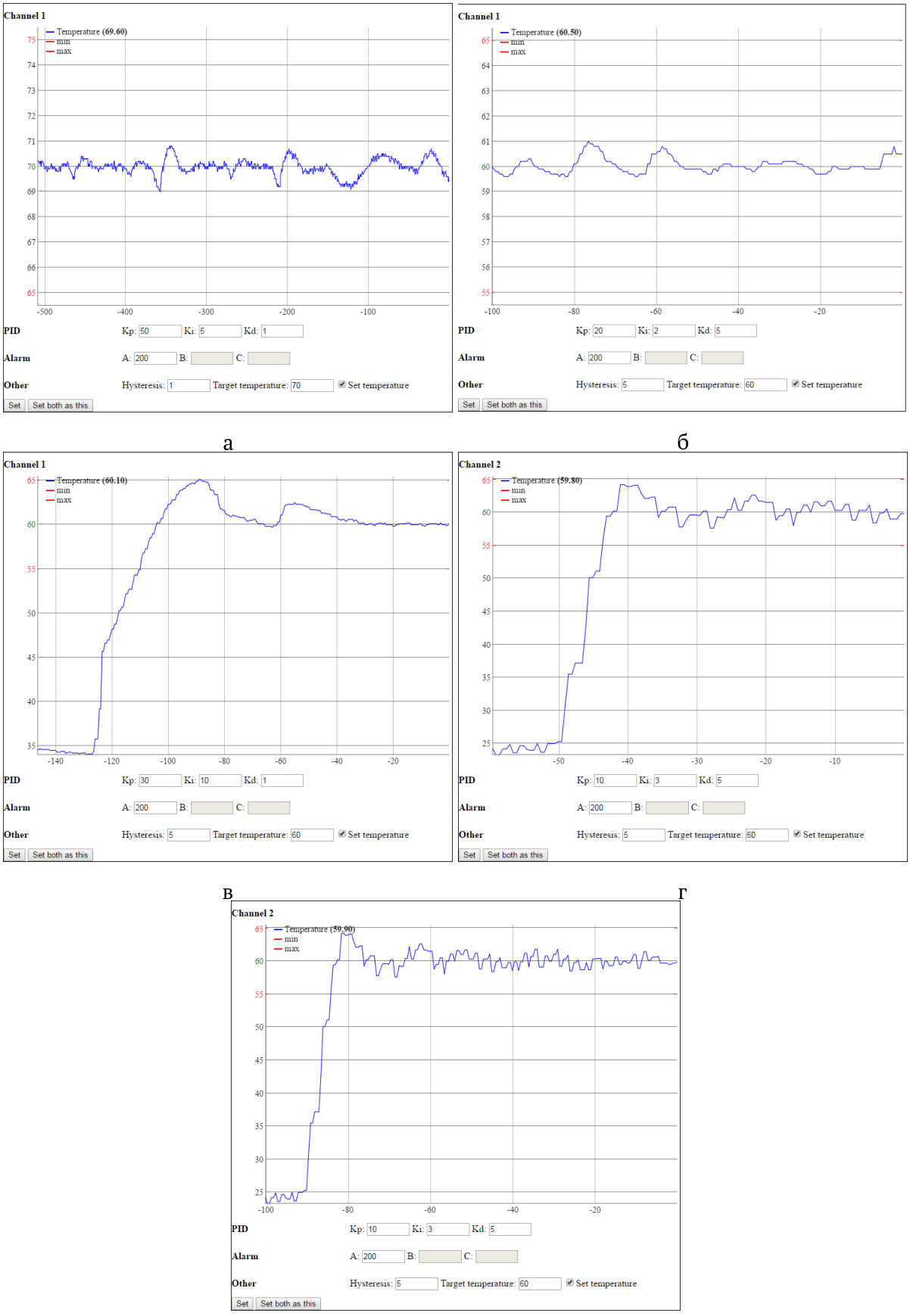


Рисунок 4.2 — Графики температуры в ходе экспериментов