1.1 Разработка двухволнового источника оптического излучения

Работы по данному направлению были проведены в два этапа.

На первом этапе экспериментально исследовались лазерные диоды различных типов и производителей. Управление лазерными диодами осуществлялось с помощью контроллеров сторонних производителей. В ходе данного этапа были измерены зависимости спектров излучения лазерных диодов от температуры и управляющего тока. На основании результатов измерений был выбран тип лазерных диодов с наибольшей стабильностью в режиме непрерывного излучения, а также более пологой зависимостью длины волны излучения от температуры и силы управляющего тока, что является существенным фактором для дальнейшего управления длиной волны лазерного излучения.

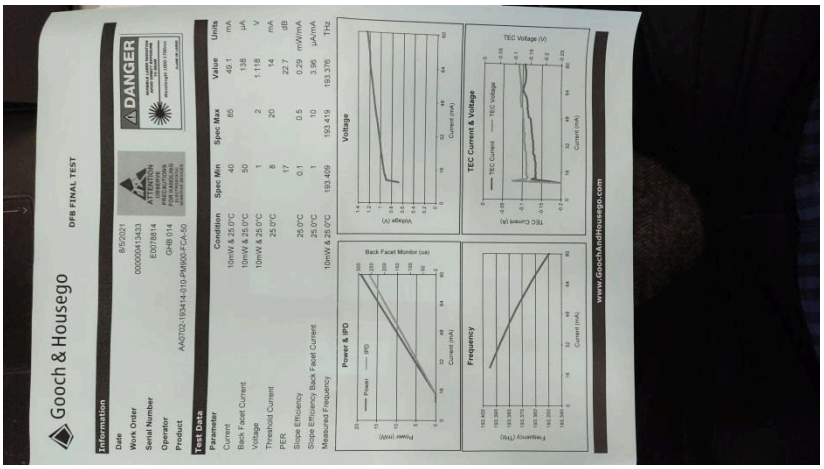
На втором этапе была разработана электронная управляющая плата и программное обеспечение для нее с графическим интерфейсом, позволяющие управлять одновременной работой двух лазерных диодов и контролировать параметры их работы в реальном времени. Данная управляющая плата была изготовлена в двух экземплярах, после чего была продемонстрирована успешная работа данных плат по управлению парами лазерных диодов российской компании Нолатех в двух диапазонах длин волн – 1550 нм и 839 нм.

Этап 1. Изучение зависимости спектров излучения лазерных диодов от температуры и тока, выбор оптимального лазерного диода

1. DFB-лазерный диод фирмы Gooch & Housego производства США марки: AA0702 193414-010-PM900-FCA50 E0078814 с внешним ВЧ входом модуляции.

Диод работал в непрерывном режиме (CW) с максимумом на длине волны 1,55 мкм. Максимальная выходная мощность лазерного диода 10 мВт. На рисунке 1.1.1.1 представлены фотографии самого лазерного диода и его краткое описание.

а)



б)

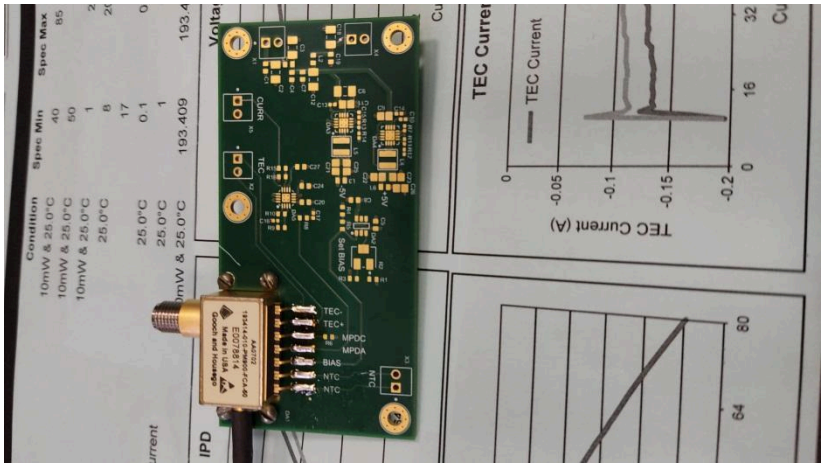


Рисунок 1.1.1.1. Лазерный диод AA0702 б) и его описание а).

Измерительный стенд был построен следующим образом: лазерный диод закреплялся на массивном алюминиевом радиаторе с использованием

термопроводящей пасты КПТ-8 и подсоединялся к термостабилизирующему блоку DX5100 с точностью измерения температуры 0,01 К, ток на диод подавался от источника тока повышенной точности АКИП-111332В/3А с точностью измерения тока 1 мкА.

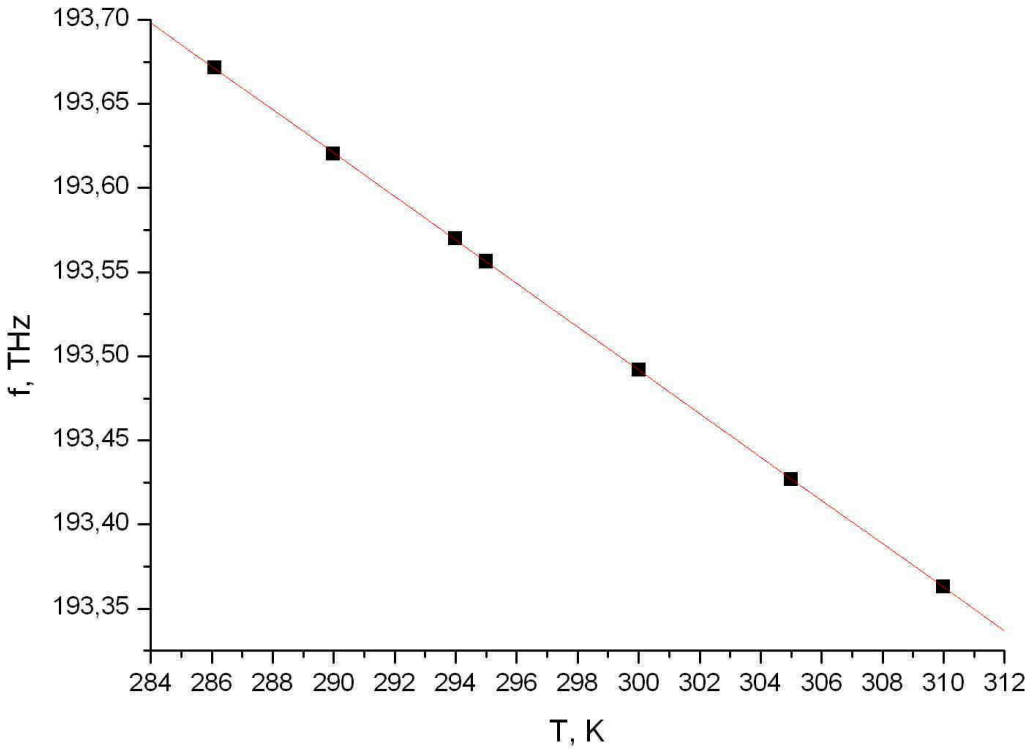


Рисунок 1.1.1.2. Зависимость максимума частоты выходного излучения лазерного диода АА0702 от температуры при постоянном токе.

На рисунке 1.1.1.2 приведена зависимость максимума частоты излучения лазера АА0702 при постоянном токе равном 22 мА. Она линейна и из ее наклона следует, что частота изменяется по закону -0,013 ТГц/К или -13 ГГц/К.

На рисунке 1.1.1.3 показана зависимость максимума частоты излучения лазера от тока при температуре 295 К или 22˚С. Она тоже линейна и изменяется по закону $-4*10^{-4}$ ТГц/мА или -0,4 ГГц/мА.

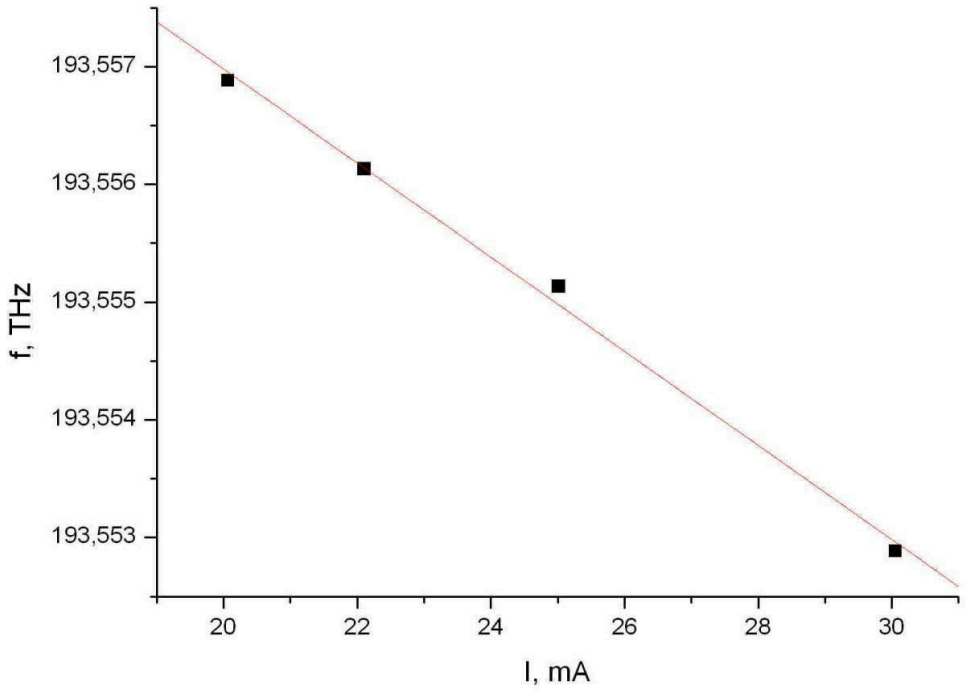


Рисунок 1.1.1.3. Зависимость максимума частоты выходного излучения лазерного диода AA0702 от тока при постоянной температуре.

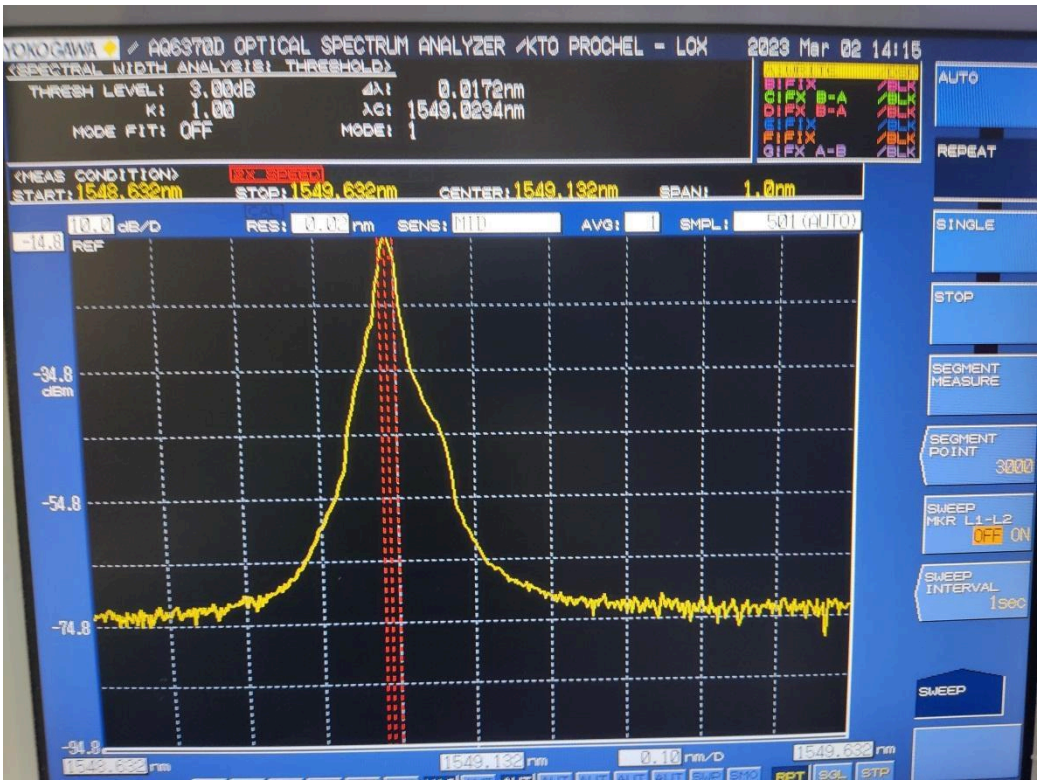


Рисунок 1.1.1.4. Общий вид спектрограммы излучения лазерного диода AA0702, развертка 0,1 нм/дел.

От лазерного диода оптический сигнал подавался непосредственно на анализатор спектра YOKOGAWA AQ6370D. Выходная мощность диода варьировалась в зависимости от подаваемого на него тока от 0,8 до 4 мВт. На рисунке 1.1.1.4 представлена фотография одного из полученных спектров излучения лазерного диода с центральной полосой 1549,132 нм. Анализатор спектра был настроен таким образом, чтобы каждое новое измерение запускалось по изменению центральной полосы сигнала. Было отмечено, что данный тип диода не достаточно стабильно держит центральную полосу спектра, и она смещается на 0,002 нм с периодичностью примерно раз в 1-2 секунды, что равно уходу приблизительно на 250 МГц.

На рисунке 1.1.1.1, а) производителем приведена зависимость центральной полосы спектра излучения диода от тока при 25˚C, что не совпадает с нашими измерениями (рисунок 3). Мы получили очень близкие значения с указанными производителем, но при температуре 34,5˚C.

2. DFB-лазерный диод китайского производства марки: SWLD-1548.51-FC/PC-05-PM.

Диод работал в режиме CW с максимумом на длине волны 1,54 мкм. Максимальная выходная мощность диода 10 мВт. Этот диод имеет корпус «бабочка» с 14 выводами и является очень близким **аналогом диода DFB-1550-14BF производства фирмы ООО «Нолатех» Россия, Москва.**

Измерительный стенд был построен следующим образом: лазерный диод закреплялся в специальной теплоотводящей подставке прибора THOR CLD1015 («маунта»), показанного на рисунке 1.1.1.5.

Рисунок 1.1.1.5. Внешний вид лицевой панели прибора THOR.

«Маунт» обеспечивал отвод тепла от лазера, стабилизацию температуры с точностью до 0,01˚С и стабилизацию тока с точностью до 0,1 мА, что можно видеть на рисунке. Способ установки лазера в прибор продемонстрирован на рисунке 1.1.1.6.

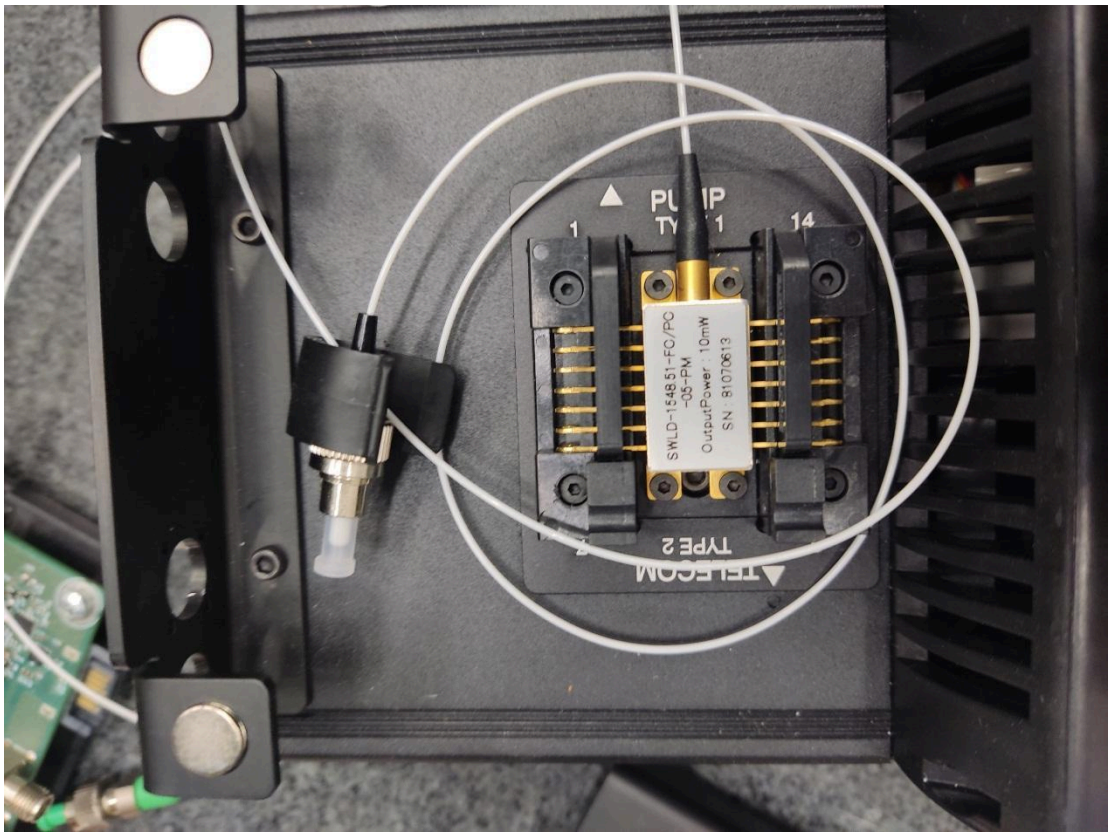


Рисунок 1.1.1.6. Внешний вид верхней части прибора THOR с установленным лазерным диодом.

От лазерного диода оптический сигнал подавался также на анализатор спектра YOKOGAWA AQ6370D.

На рисунке 1.1.1.7 представлена зависимость максимума частоты излучения лазера SWLD-1548.51 при постоянном токе равном 22 мА. Она линейна и из ее наклона следует, что частота изменяется по закону -0,014ТГц/˚С или -14 ГГц/˚С, что практически полностью совпадает с наклоном зависимости, отснятой для диода АА0702.

На рисунке 1.1.1.8 приведена зависимость максимума частоты излучения лазера от тока при температуре 293 К или 20˚С. Она тоже линейна и изменяется по закону -0,0017 ТГц/мА или -1,7 ГГц/мА.

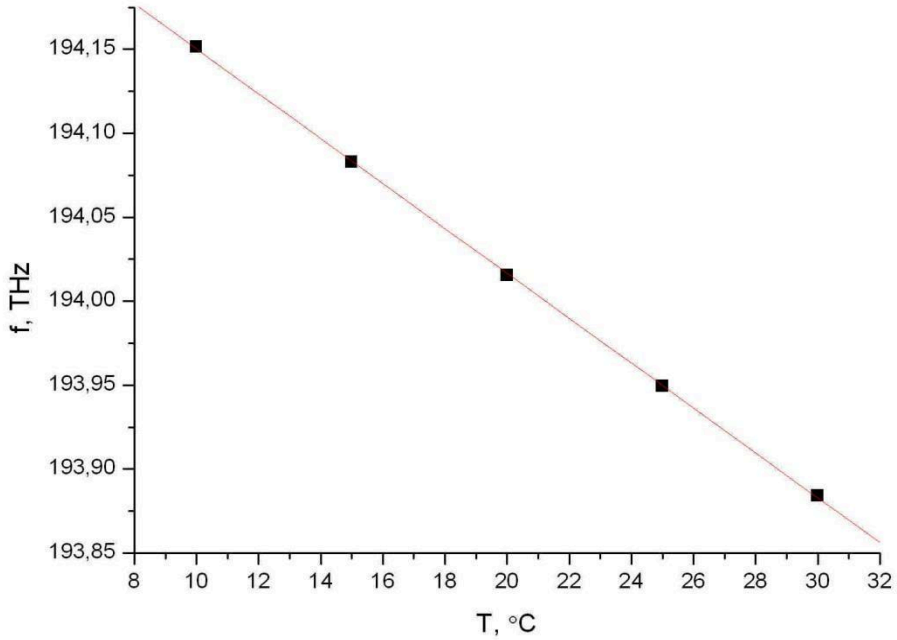


Рисунок 1.1.1.7. Зависимость максимума частоты выходного излучения лазерного диода SWLD-1548.51 от температуры при постоянном токе.

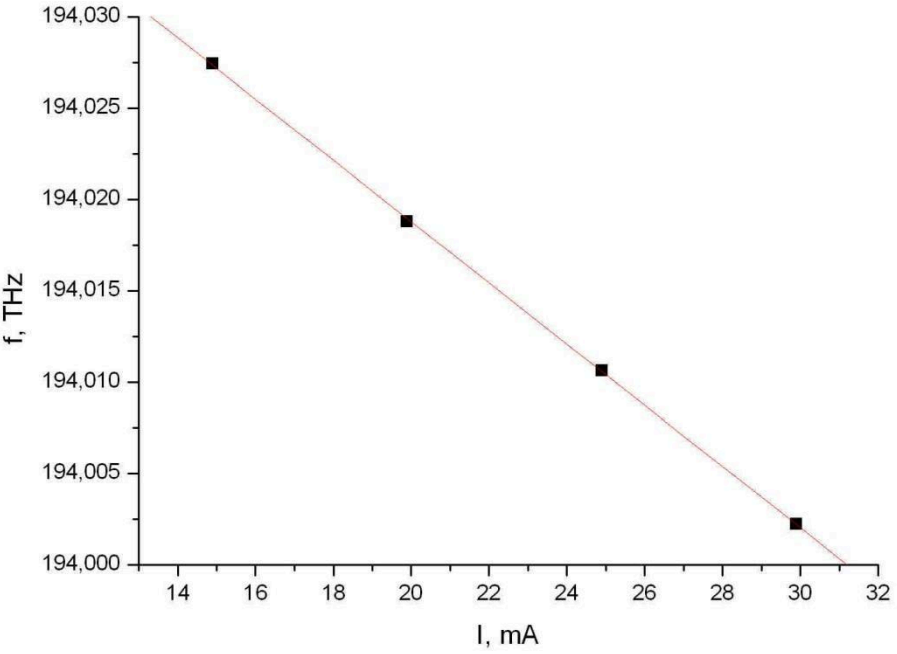


Рисунок 1.1.1.8. Зависимость максимума частоты выходного излучения лазерного диода SWLD-1548.51 от тока при постоянной температуре.

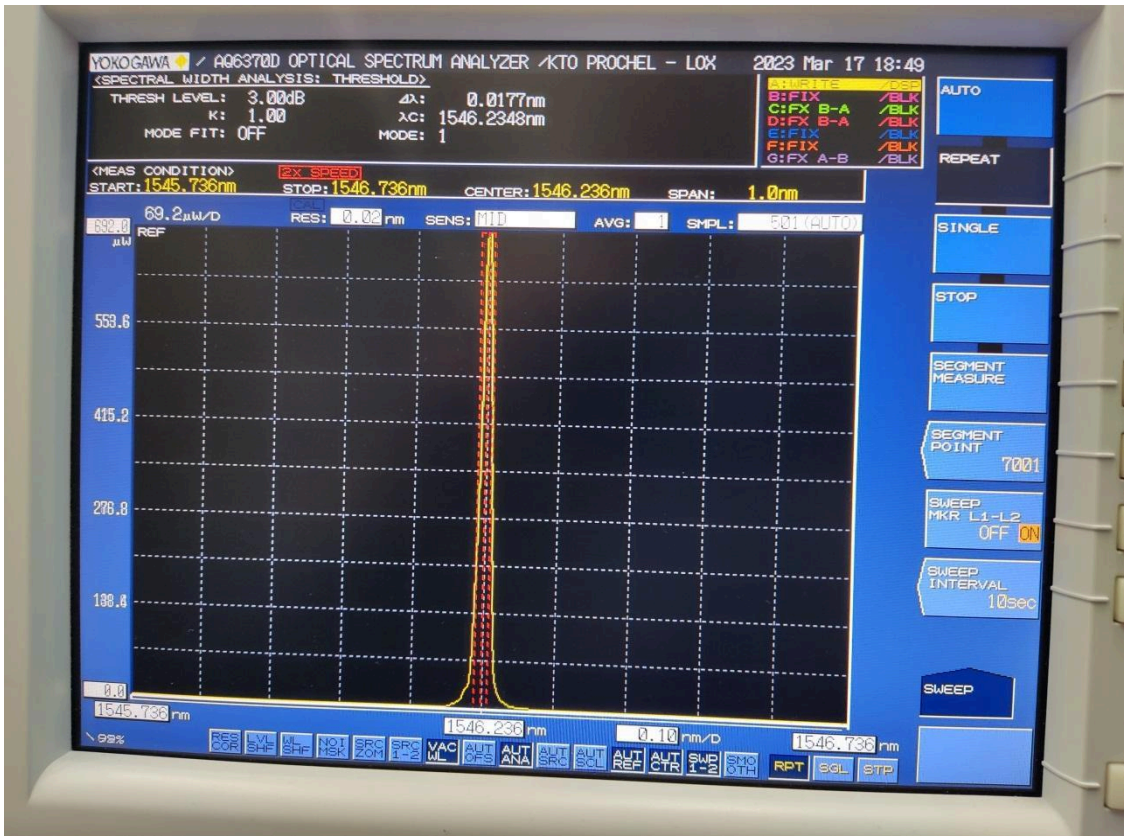


Рисунок 1.1.1.9. Спектр излучения лазерного диода SWLD-1548.51 при 20˚C и токе 29,9 мА, развертка 0,1 нм/дел.

На рисунке 1.1.1.9 показан спектр излучения лазерного диода SWLD-1548.51. Данный диод, работая в режиме CW, показал более стабильные результаты по удержанию центральной полосы, чем диод AA0702, и она смещалась на 0,002 нм с периодичностью примерно раз в 10-15 секунды,

3. Лазерный диод с резонатором Фабри-Перо, с внешней брэгговской решеткой BLD-1310-14BF-10mW производства фирмы ООО «Нолатех» Россия, Москва.

Диод работал в режиме CW с максимумом на длине волны 1,31 мкм. Максимальная выходная мощность диода 10 мВт. Этот диод имеет корпус «бабочка» с 14 выводами. Диод интересен тем, что имеет внешнюю брэгговскую решетку, нанесенную в самом начале оптоволокна, не связанную с самим кристаллом и вырезающим узкую область спектра излучения диода шириной не более 100 кГц.

Измерительный стенд был построен следующим образом: лазерный диод закреплялся в специальной теплоотводящей подставке прибора THOR CLD1015.

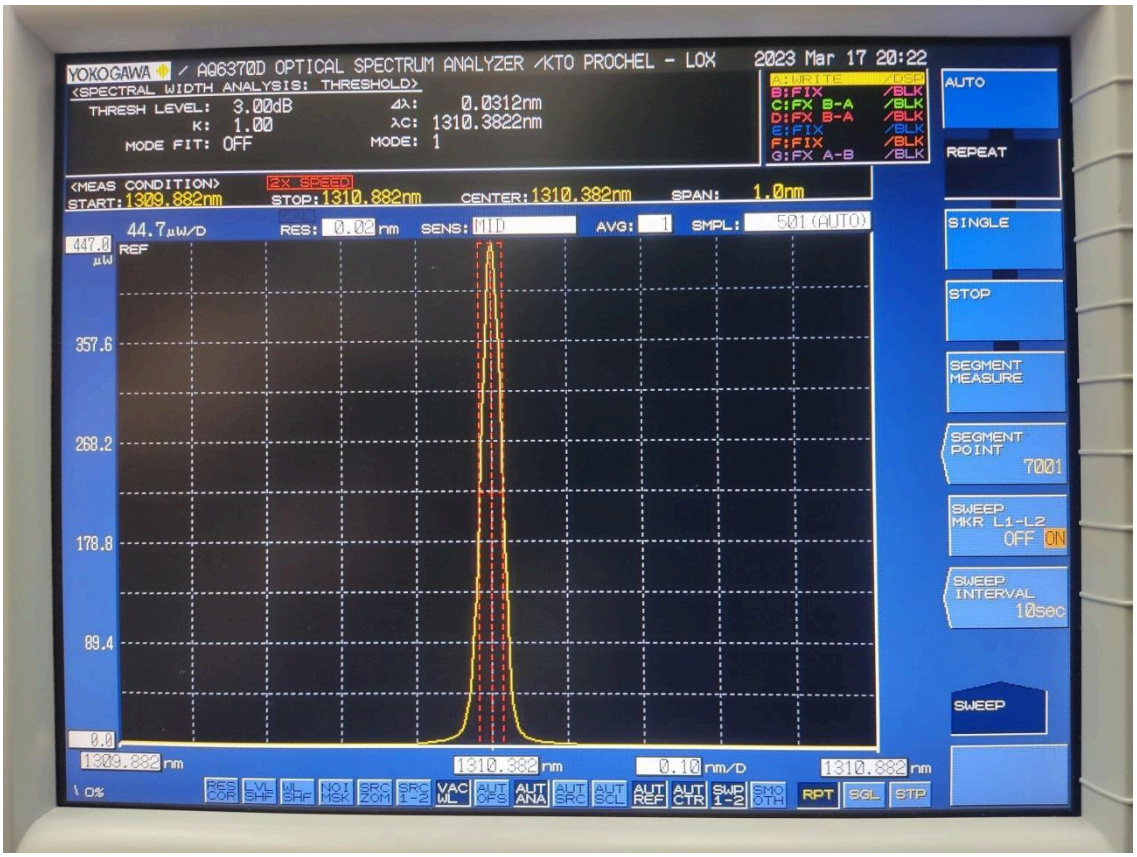
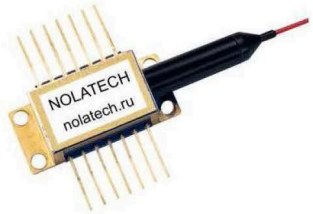


Рисунок 1.1.1.10. Спектр излучения лазерного диода BLD-1310-14BF при 20˚C и токе 29,9 мА, развертка 0,1 нм/дел.

Russia, Moscow
Tel. \ fax: (495) 333-93-01
www.nolatech.ru
nolatech@mail.ru

NOLATECH

Laser Diode 1310nm 20mW



BLD-1310-14BF Fiber Bragg Grating laser is single frequency laser diode module designed for optical measurement and communication. The laser is packaged in 14-pin standard butterfly package with monitor photodiode and thermo-electric cooler (TEC).

Key Features

- Optical output: 20mW
- Narrow linewidth (Δv < 0.1MHz)
- Wavelength: 1310nm @ 25°C
- SM or PM Fiber (ø0.9mm)
- FC-APC connector
- 14-pin butterfly package
- Internal monitor PD and TEC
- Low power consumption

Optical and electrical characteristics: (T = 25°C)

Item	Symbol	Test condition	Min.	Typ.	Max.	Unit
Output Power	P_f			20	30	mW
Forward Voltage	V_F	P_f=20mW			2.5	V
Threshold Current	I_{th}			30	50	mA
Forward Current	I_F	P_f=20mW		200	300	mA
Center Wavelength	λ_c	P_f=20mW	1270		1330	nm
Spectral Width	Δλ	P_f=10mW		100		kHz
Side Mode Suppression Ratio	SMSR	P_f=20mW	40	45		dB
Monitor Current	I_m	P_f=20mW, V_{RD}=5V	40		200	μA
PD Dark Current	I_d	V_{RD}=5V			0.1	μA
Cooler Voltage	V_C	I_F=EOL, TC=70°C			2.7	V
Cooler Current	I_C	I_F=EOL, TC=70°C			1.4	A
Thermal Resistance	R_o	T_{LD}=25°C, B=3900±100K	9.5	10.0	10.5	kΩ
Extinction Ratio	X_P	P_f=20mW	17			dB
Mode Hop Free Range	ΔI			30		mA
Single-Frequency Continuous Tuning Range	Δf		3			GHz
Current Tuning	Δλ/ΔI			0.002		nm/mA
Temperature Tuning	Δλ/ΔT			0.08		nm/°C

Рисунок 1.1.1.11. Общие оптические и электрические характеристики лазерного диода BLD-1310-14BF, заявленные ООО «Нолатех».

На рисунке 1.1.1.10 представлен общий вид спектра лазерного диода BLD-1310-14BF. Диод работал в режиме CW и при токах I>=30 мА стабильно

удерживал центральную полосу спектра, практически без колебаний. Следует отметить, что при токе I>=30 мА линия становилась тонкой, а при более низких токах (когда диод ещё не полностью разгорелся) спектральная линия была более широкой и нестабильной.

На рисунке 1.1.1.11 показана первая страница описания лазерного диода BLD-1310-14BF ООО «Нолатех» из которой следует, что нормальный ток работы диода в режиме CW составляет 30 мА, а максимальный 50 мА. Исходя из этих данных, была получена зависимость максимума частоты выходного излучения от температуры при постоянном токе равном 29,9 мА, приведенная на рисунке 12. Она линейна и из ее наклона следует, что частота изменяется по закону -0,12 ТГц/˚С.

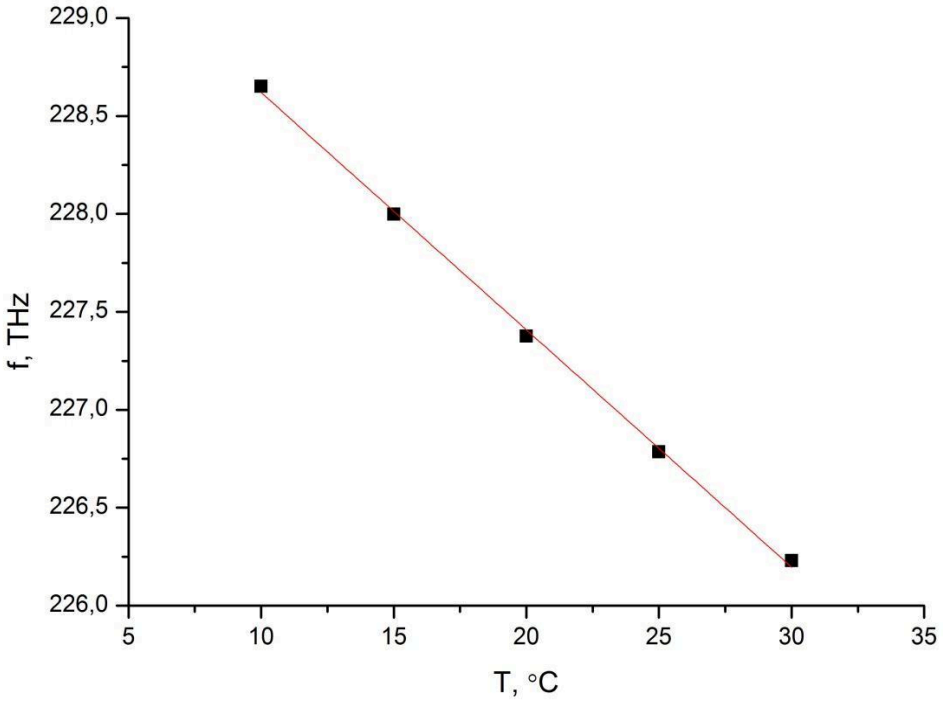


Рисунок 1.1.1.12. Зависимость максимума частоты выходного излучения лазерного диода BLD-1310-14BF от температуры при постоянном токе.

На рисунке 1.1.1.13 представлена зависимость максимума частоты выходного излучения от тока при постоянной температуре равной 20˚С. Она близка к линейной и из ее наклона следует, что частота изменяется по закону 0,127 ТГц/мА.

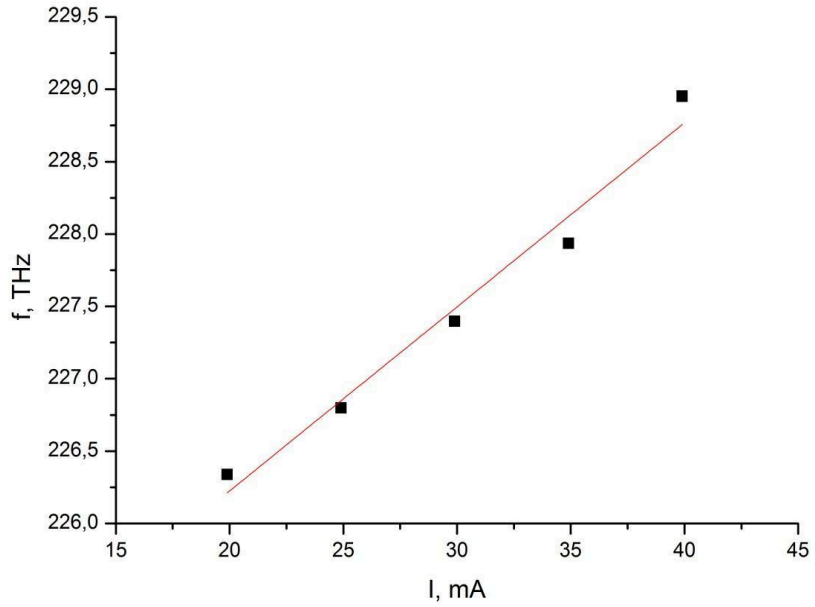


Рисунок 1.1.1.13. Зависимость максимума частоты выходного излучения лазерного диода BLD-1310-14BF от тока при постоянной температуре.

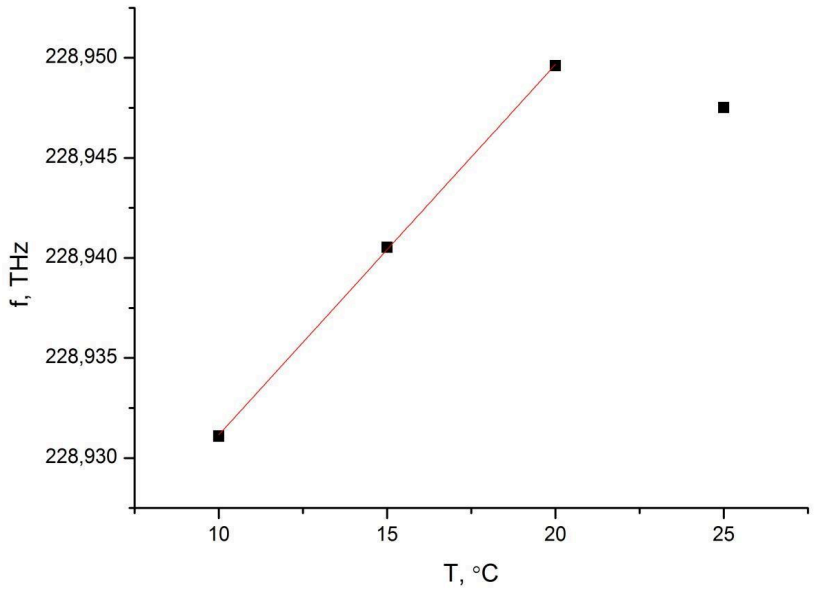


Рисунок 1.1.1.14. Зависимость максимума частоты выходного излучения лазерного диода BLD-1310-14BF от температуры при постоянном токе (I=39,9 mA).

Из наблюдений за разгоранием лазерного диода такого типа было отмечено, что ток равный 30 мА действительно является для него неким пороговым значением. При достижении которого, лазер переходит в стабильный режим удержания центральной полосы спектра, линия спектра обретает вид узкой одномодовой линии, показанной на рисунке 10. В связи с чем, было принято решение провести исследование зависимости максимума частоты выходного излучения от температуры при постоянном токе превышающим пороговое значение, а именно при токе равном 39,9 мА. Полученная таким образом зависимость изображена на рисунке 14. Зависимость частично линейна и имеет наклон, изменяющийся по закону 1,85 ГГц/˚С. Из которой видно, что для данного типа лазерного диода с пороговым током 30 мА можно отметить, что одномодовый режим сохраняется до температуры 25˚С в CW режиме. При этом частота излучения моды увеличивается с увеличением температуры. Дальнейшее увеличение температуры приводит к уменьшению ее интенсивности и генерации соседних длинноволновых мод, делая лазер многомодовым. Наиболее интенсивной становится самая длинноволновая мода или соседняя с ней, отстоящая на 3−5 межмодовых расстояний от моды, генерировавшейся при меньших температурах. Многомодовый режим излучения лазерного диода при температуре 25˚С продемонстрирован на рисунке 1.1.1.15.

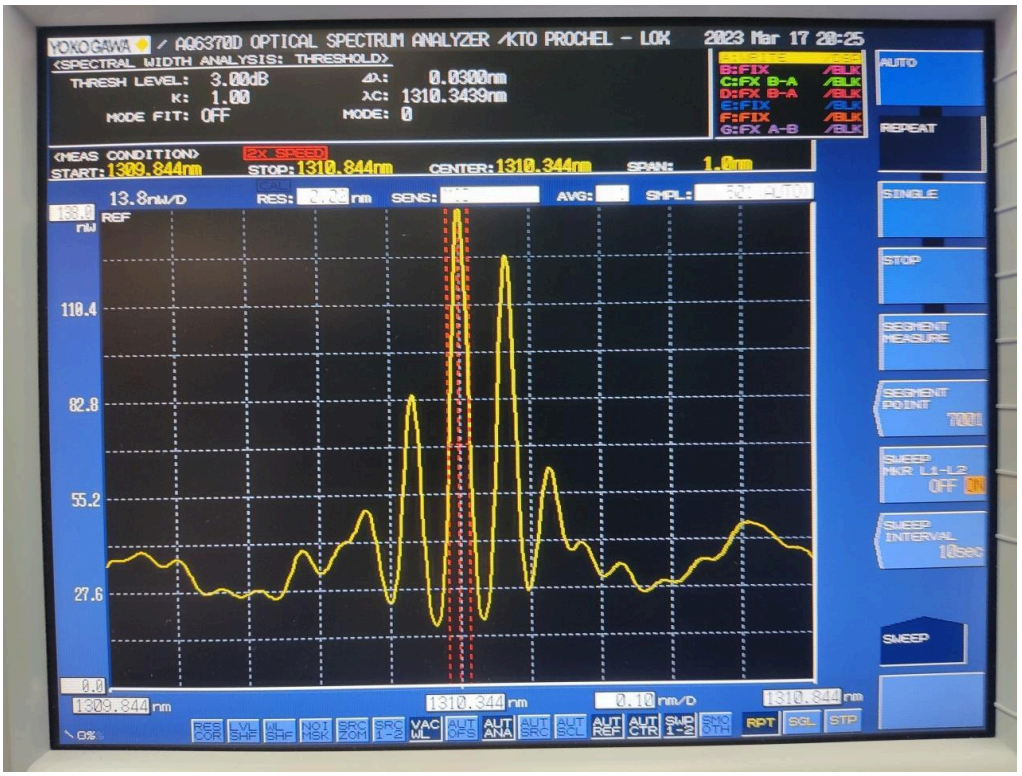


Рисунок 1.1.1.15. Спектр излучения лазерного диода BLD-1310-14BF при 25˚C и токе 39,9 мА, развертка 0,1 нм/дел.

Выводы

Из исследованных лазерных диодов **самым стабильным в режиме CW оказался диод китайского производства SWLD-1548.51, у него же наблюдались и наименьшая крутизна зависимостей центральной полосы спектра, как от температуры, так и от тока.**

Спектральная ширина DFB-диодов в CW режиме намного уже, чем у лазера Фабри-Перо, даже оснащенного брэгговской решеткой. Крутизна зависимости центральной полосы спектра от температуры у DFB-диодов на порядок лучше, а от тока лучше примерно на два порядка.

У лазерного диода с резонатором Фабри-Перо и внешней брэгговсой решеткой наблюдается обратная зависимость центральной полосы спектра от

тока, возможно, это связано с тем, что основным механизмом модуляции длины волны излучения в данном случае является зависимость коэффициента преломления от температуры. При превышении тока над пороговым значением примерно на 50% наблюдаются нелинейные оптические взаимодействия мод резонатора, приводящие к многомодовому режиму генерации.

Этап 2. Разработка и изготовление двухволнового источника оптического излучения

Плата электронная управляющая

Для того, чтобы установить и поддерживать полупроводниковые лазерные диоды в необходимом режиме, был создан драйвер лазеров. Плата позволяет задавать необходимый ток прямого смещения через лазерный диод, измерять и удерживать температуру кристалла, а также измерять относительную выходную мощность. 3D-модель электронной управляющей платы представлена на рисунках 1.1.2.1 и 1.1.2.2.

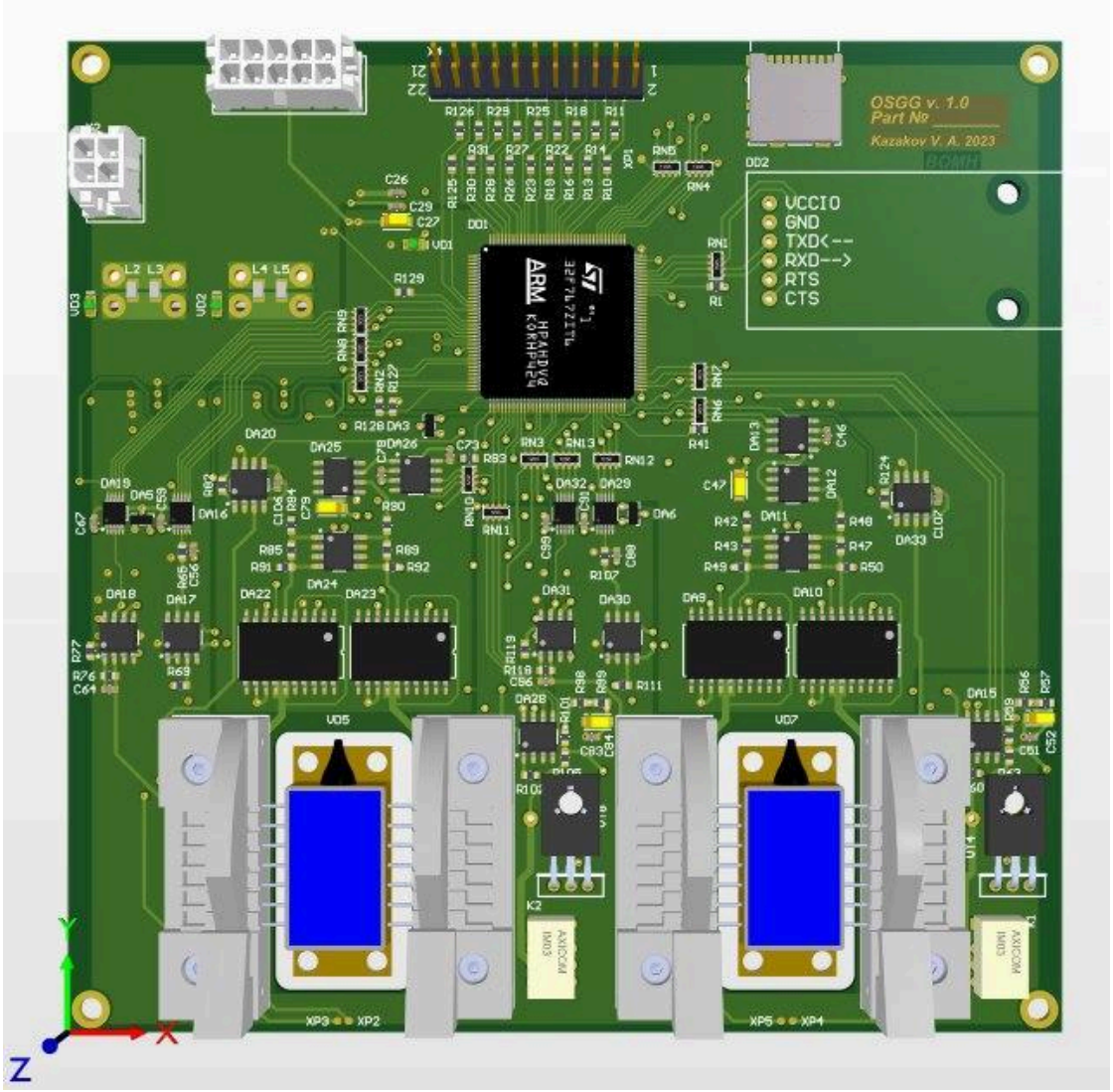


Рисунок 1.1.2.1. 3D-модель электронной управляющей платы, вид сверху.

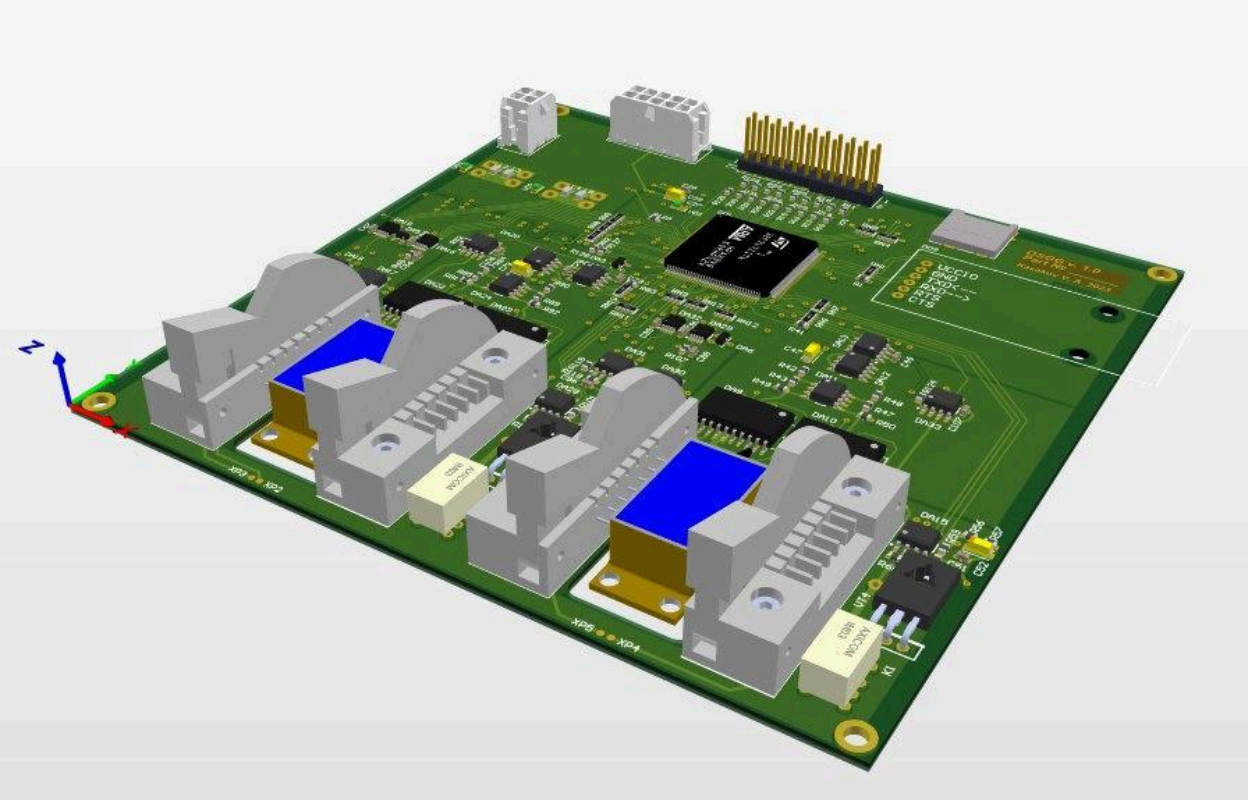


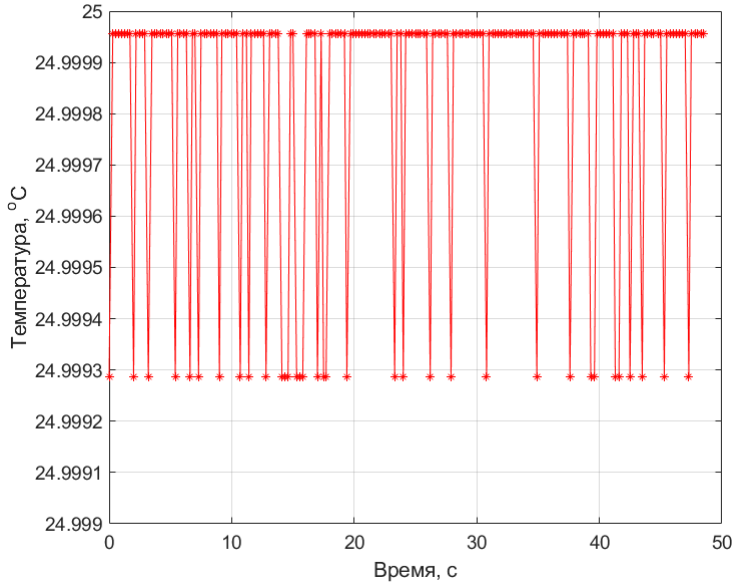
Рисунок 1.1.2.2. 3D-модель электронной управляющей платы, вид сбоку.

Управляющей микросхемой устройства является микроконтроллер STM32F767ZIT6 фирмы «STMicroelectronics». Достаточное быстродействие микросхемы (частота центрального процессора Cortex-M7 до 216 МГц) позволяет одновременно управлять двумя лазерными модулями с частотой 10 Гц. Микроконтроллер принимает команды и отправляет данные через USB-порт благодаря встроенному в плату модулю CP2102 USB UART Board фирмы «Waveshare».

Для задания тока лазера используется управляемый источник тока. Перестройка тока для лазеров TLD-840-14BF составляет от 0 до 200 мА с шагом 3 мкА, для лазеров DFB-1550-14BF – от 0 до 70 мА с шагом 1 мкА. При этом можно задавать ток лазера, периодически изменяющийся во времени. Частота периодического сигнала составляет 10 Гц при 100 точках на один период. Форма каждого импульса произвольная.

Измерение температуры кристалла осуществляется при помощи встроенного в модуль термистора и схемы усиления и оцифровки сигнала внутреннего термистора. Сочетание точности термистора, малых собственных шумов схемы и шумовой фильтрации сигнала (активный НЧ-фильтр и цифровое усреднение) позволяет измерять температуру кристалла лазера в диапазоне от -1,3 до 45,9 °C с погрешностью измерения $\pm 0{,}35 \cdot 10^{-3}$ °C.

Внутренняя температура каждого лазерного модуля не только измеряется, но и поддерживается на заданном значении благодаря встроенному элементу Пельтье и схеме его управления. На рисунке 1.1.2.3 представлена осциллограмма полученных значений температуры. Установленная температура в данном примере равна 25 °C. Видно, что погрешность поддержания температуры соответствует погрешности измерения ($0{,}35 \cdot 10^{-3}$ °C).



Рисунко 1.1.2.3. Осциллограмма термостабилизации лазерного модуля.

Для учёта температурного дрейфа системы в схеме предусмотрен контроль внешней температуры в непосредственной близости от лазерных модулей.

Измерение внешней температуры осуществляется при помощи термистора B57330V2103F260 фирмы «TDK» и схемы усиления и оцифровки сигнала внешнего термистора. Диапазон измерения температуры составляет от -25.78 до 43.36 °C с погрешностью измерения ±0,02 °C.

Каждый из используемых лазерных модулей содержит мониторный фотодиод, фототок которого прямо пропорционален мощности лазерного излучения. В состав устройства входят каскады усиления и оцифровки фототока мониторных фотодиодов. Устройство позволяет измерять фототок в диапазоне от 0 до 500 мкА. При этом погрешность оцифровки составляет ±4 нА. Таким образом, зная чувствительность мониторного фотодиода, можно измерить мощность лазерного излучения с достаточно большой точностью.

При автономном режиме работы устройства бывает важно регистрировать параметры лазерных модулей и системы в целом. Кроме того, должна существовать возможность самостоятельной работы устройства сразу после включения питания без какого-либо внешнего управления. Для этих целей используется внешний накопитель типа microSD. Файловая система микроконтроллера позволяет использовать внешние флэш-накопители объёмом до 32 GB для хранения получаемых данных о системе. Кроме того, устройство записывает последние используемые параметры настроек и при отсутствии подключения по USB запускает эти настройки сразу после включения питания.

Фотография изготовленной платы вместе с установленными лазерными диодами представлена на рисунке 1.1.2.4.

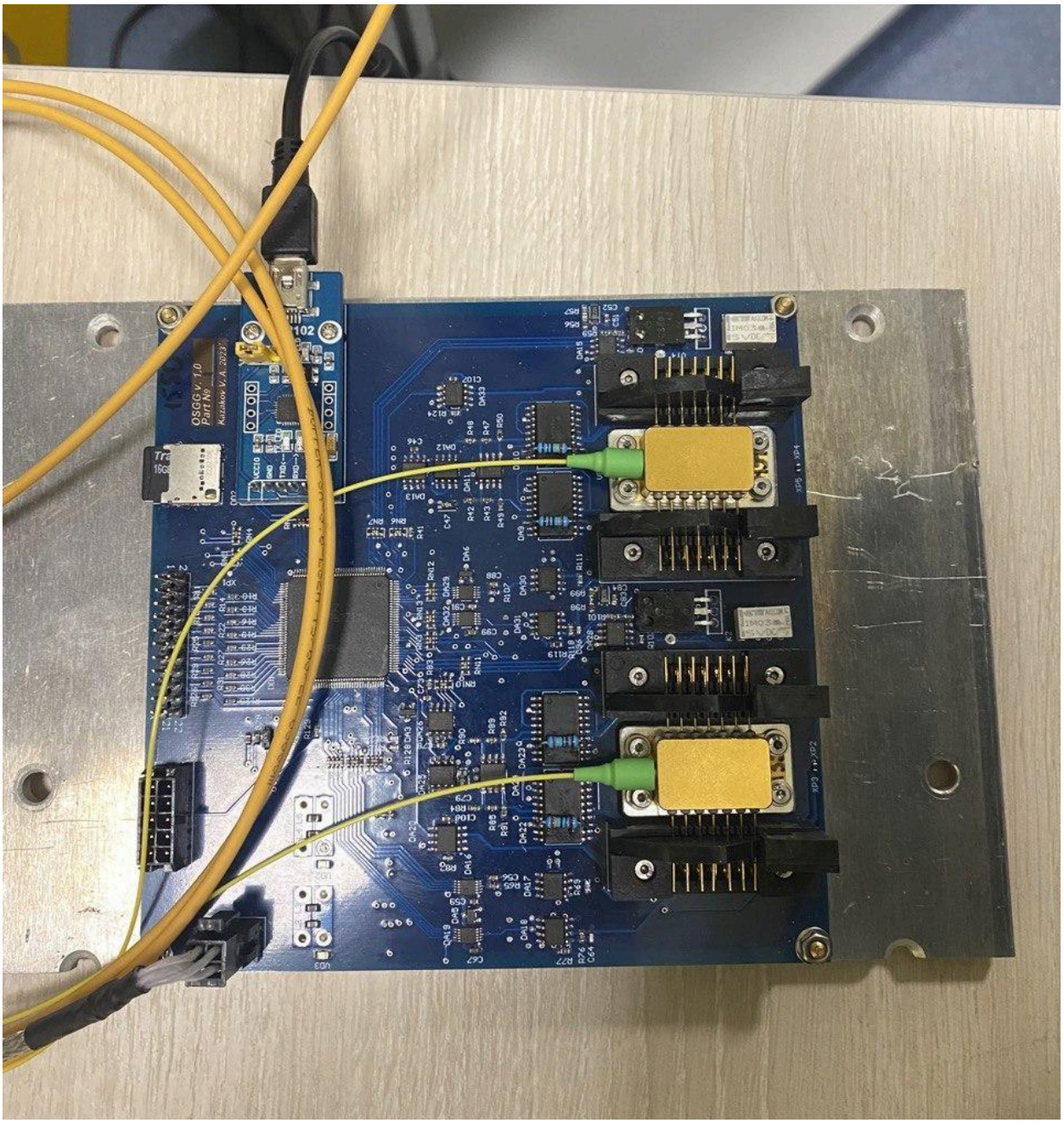


Рисунок 1.1.2.4. Фотография изготовленной электронной управляющей платы с установленными лазерными диодами.

Схема электрическая принципиальная на электронную управляющую плату приведена в Приложении 1.1.2.1. Параметры протокола взаимодействия платы с компьютером описаны в Приложении 1.1.2.2.

Программное обеспечение управления лазерными диодами

Программное обеспечение управления лазерными диодами написано для ПК (персонального компьютера) с операционной системой Windows на языке программирования Python. Взаимодействие ПК и электронной управляющей платы производится по последовательному асинхронному интерфейсу, логически совместимому с интерфейсом UART. Управляющая программа разделена на 5 логических блоков:

- device_main.py – основной управляющий модуль;
- device_interaction.py – модуль высокоуровневых команд взаимодействия электронной управляющей платы и ПК (установка соединения, сброс настроек в состояние по умолчанию, запрос состояния платы, отправка управляющих параметров на плату, чтение контрольных параметров с платы);
- device_commands.py – модуль команд низкого уровня по взаимодействию электронной управляющей платы и ПК. Высокоуровневые команды из модуля device_interaction.py состоят из набора команд низкого уровня, запрограммированных в данном модуле.
- device_conversion.py – вспомогательный модуль пересчета физических величин в целочисленный диапазон 0 – 65535, воспринимаемый платой, а также обратные преобразования;
- gui.py – модуль создания и обновления отображения графического интерфейса программы.

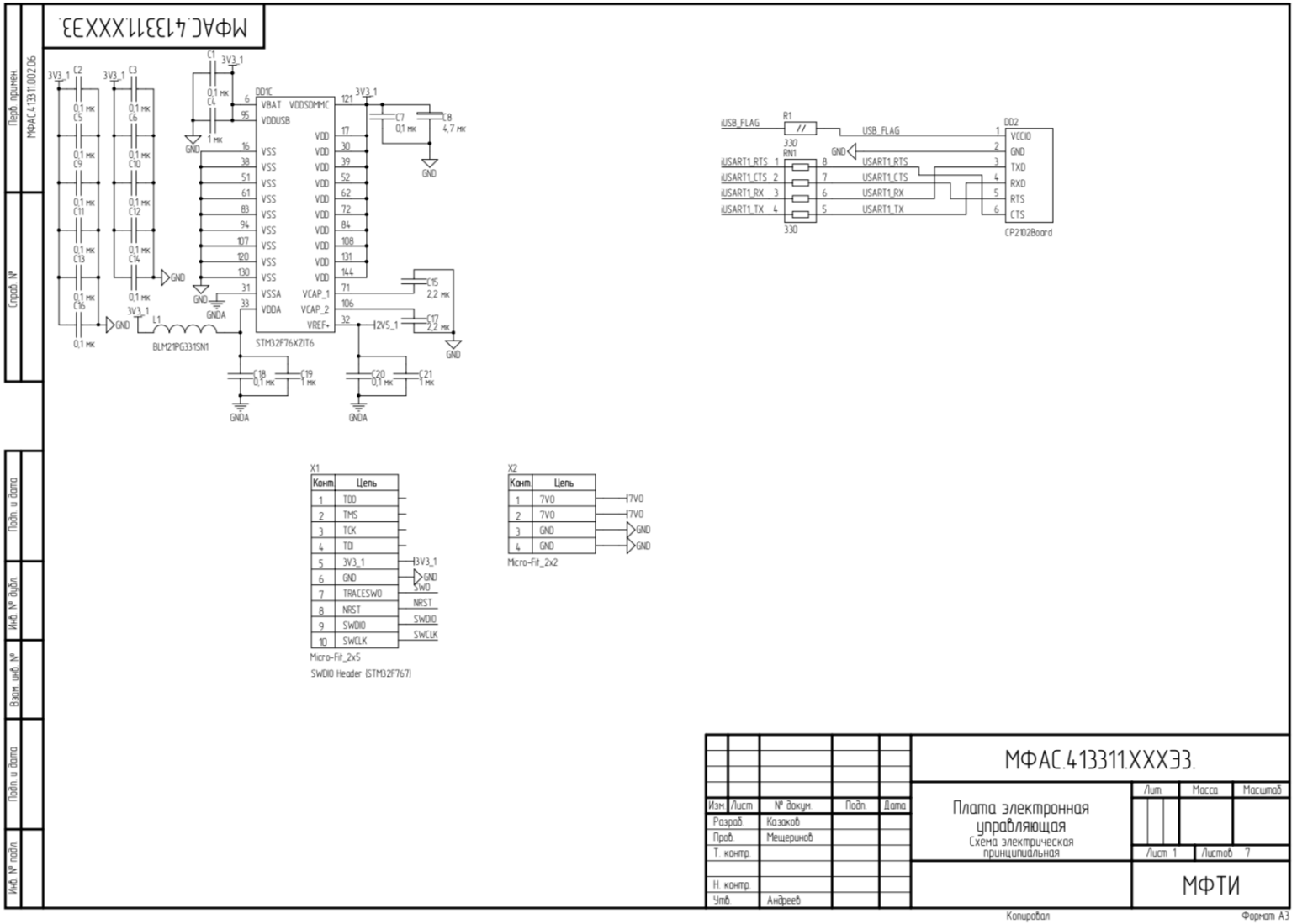
Графический интерфейс компьютерной программы управления представлен на рисунке 1.1.2.5.

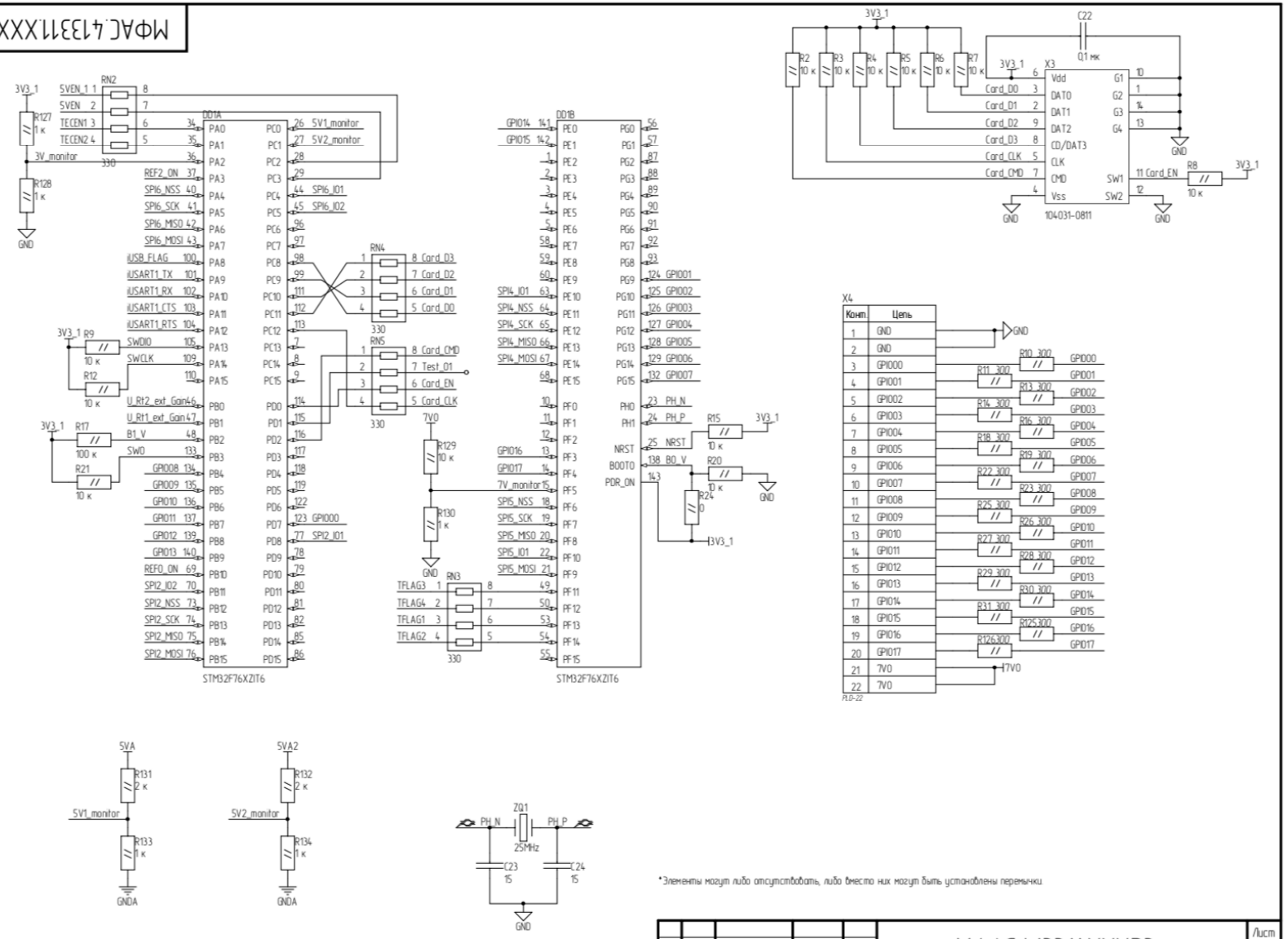
Рисунок 1.1.2.5. Графический интерфейс компьютерной программы управления лазерными диодами.

Основной графический интерфейс состоит из двух идентичных частей – левой и правой, каждая их которых относится к управлению одним из двух лазерных диодов. Текстовые поля в верхней части экрана позволяют задать управляющий ток лазерного диода в мА, а также температуру соответствующего элемента Пельтье в градусах Цельсия. Под текстовыми полями располагаются два графика, на которых отображается текущая температура лазерного диода, а также ток контрольного фотодиода. В нижней части экрана отображаются контрольные температуры и напряжения в различных частях платы.

Листинг программы управления лазерными диодами на языке Python представлен в приложении 1.1.2.3.

Приложение 1.1.2.1 Схема электрическая принципиальная на Плату электронную управляющую





					МФАС.413311.XXX33.	Лист 2
Изм.	Лист	№ докум.	Подп.	Дата		

Копировал

Формат А3

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

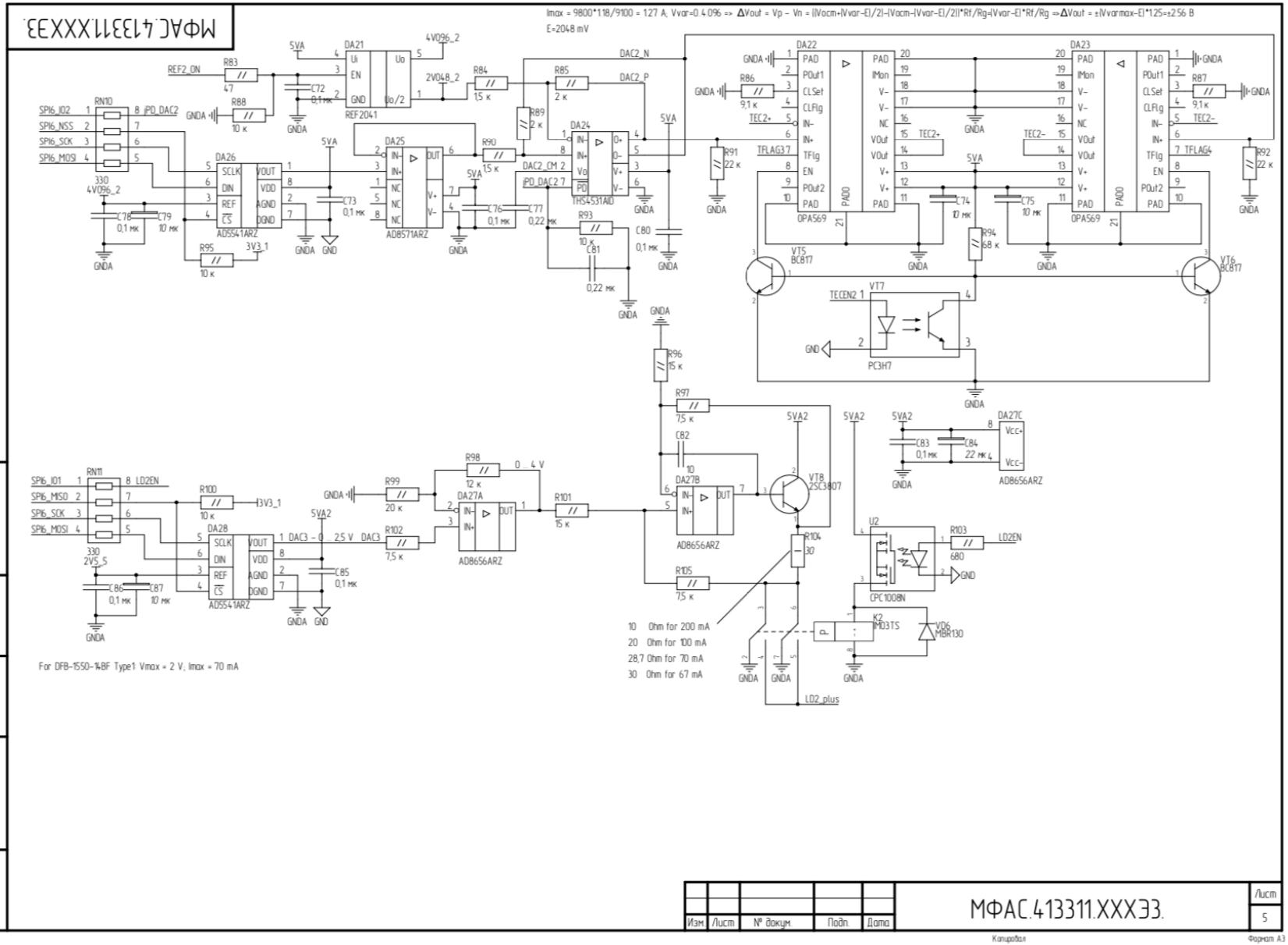
МФАС.413311.XXX33.

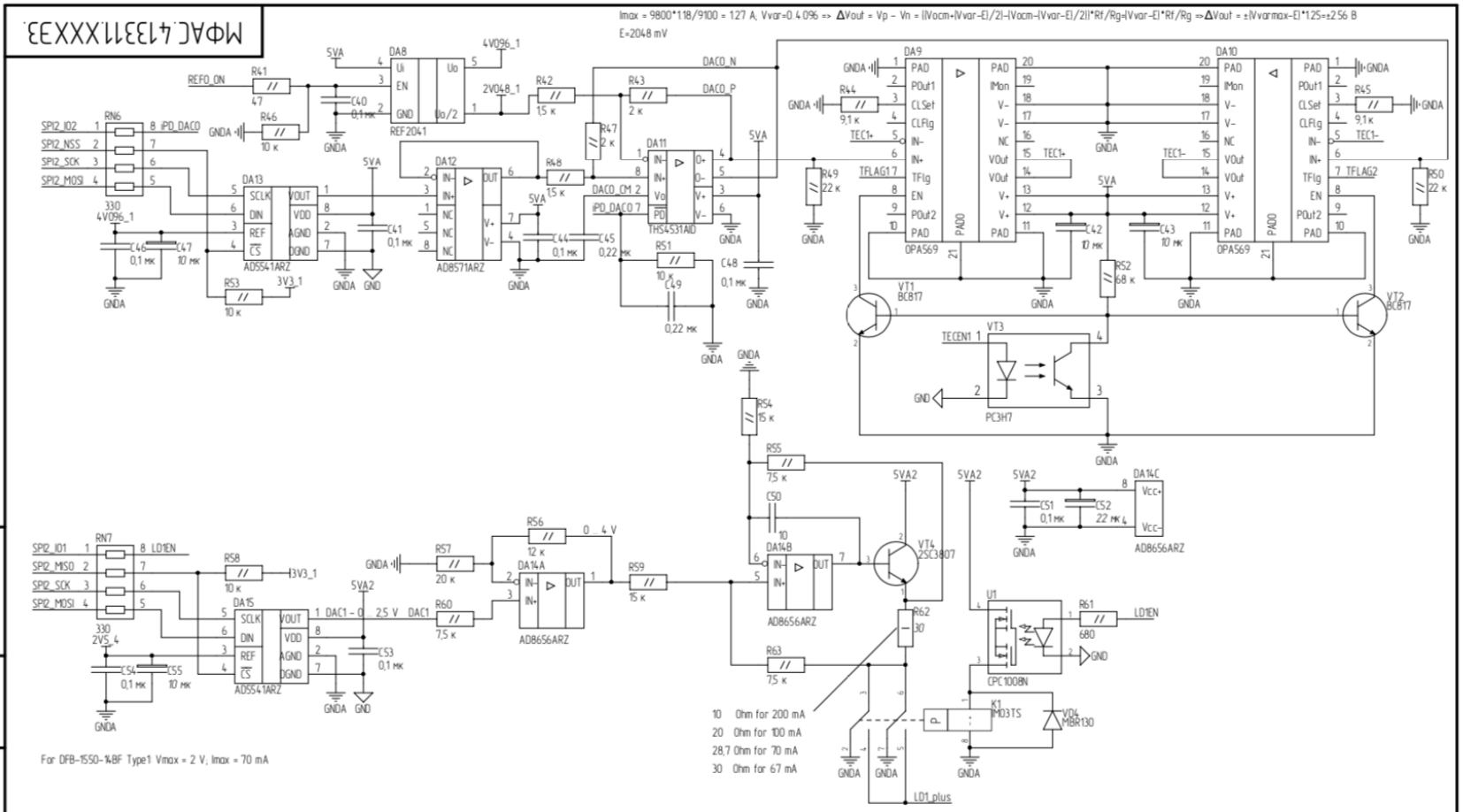
Подп. и дата | Инв. № подл. | Взам. инв. № | Инв. № дубл. | Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата	МФАС.413311.XXX33.	Лист 3

Копировал

Формат А3


Imax = 9800*1.18/9100 = 1.27 A; Vvar=0.4.096 => ΔVout = Vp – Vn = (|Vocm+|Vvar–E|/2|–|Vocm–(Vvar–E)/2|)*Rf/Rg=(Vvar–E)*Rf/Rg =>ΔVout = ±|Vvarmax–E|*1.25=±2.56 В
E=2.048 mV
DA21
REF2041
DA22
DA23
OPA569
DA24
THS4531AID
DA25
AD8571ARZ
DA26
AD5541ARZ
DA27A
DA27B
DA27C
AD8656ARZ
DA28
VT5 BC817
VT6 BC817
VT7
PC3H7
VT8 2SC3807
U2
CPC1008N
K2 MD3TS
VD6 MBR130
RN10
RN11
SPI6_IO2
SPI6_NSS
SPI6_SCK
SPI6_MOSI
SPI6_MISO
SPI6_IO1
REF2_ON
DAC2_N
DAC2_P
DAC2_OM_2
iPD_DAC2
TFLAG3_7
TFLAG4
TEC2+
TEC2–
TECEN2_1
LD2EN
LD2_plus
4V096_2
2V048_2
2V5_5
3V3_1
5VA
5VA2
GNDA
GND
0 .. 4 V
DAC3 – 0 .. 2.5 V
10 Ohm for 200 mA
20 Ohm for 100 mA
28,7 Ohm for 70 mA
30 Ohm for 67 mA
For DFB-1550-14BF Type1 Vmax = 2 V; Imax = 70 mA
МФАС.413311.XXX33.
МФАС.413311.XXX33
Изм. Лист № докум. Подп. Дата
Лист 5
Копировал
Формат А3
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.



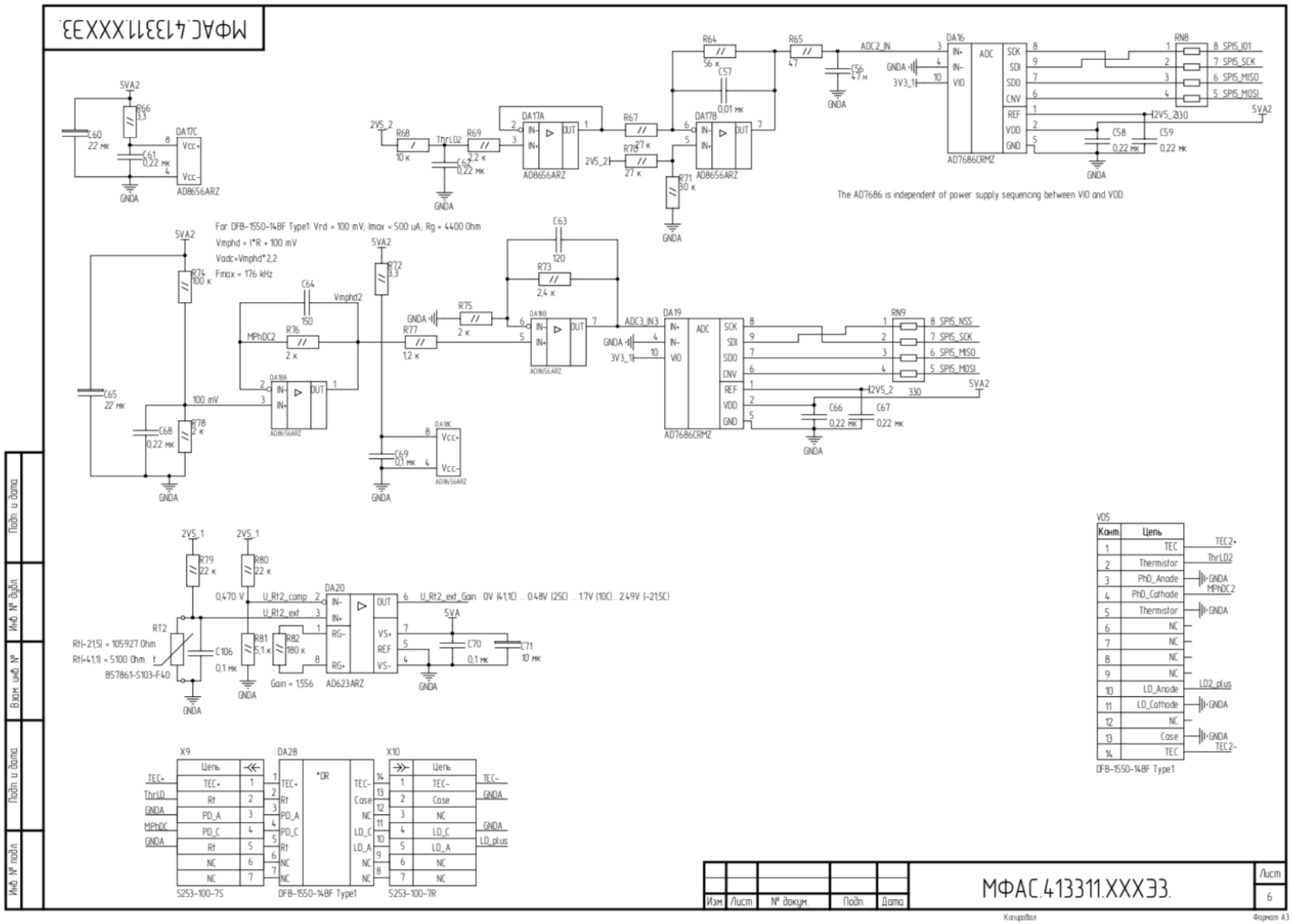
МФАС.413311.XXX3.

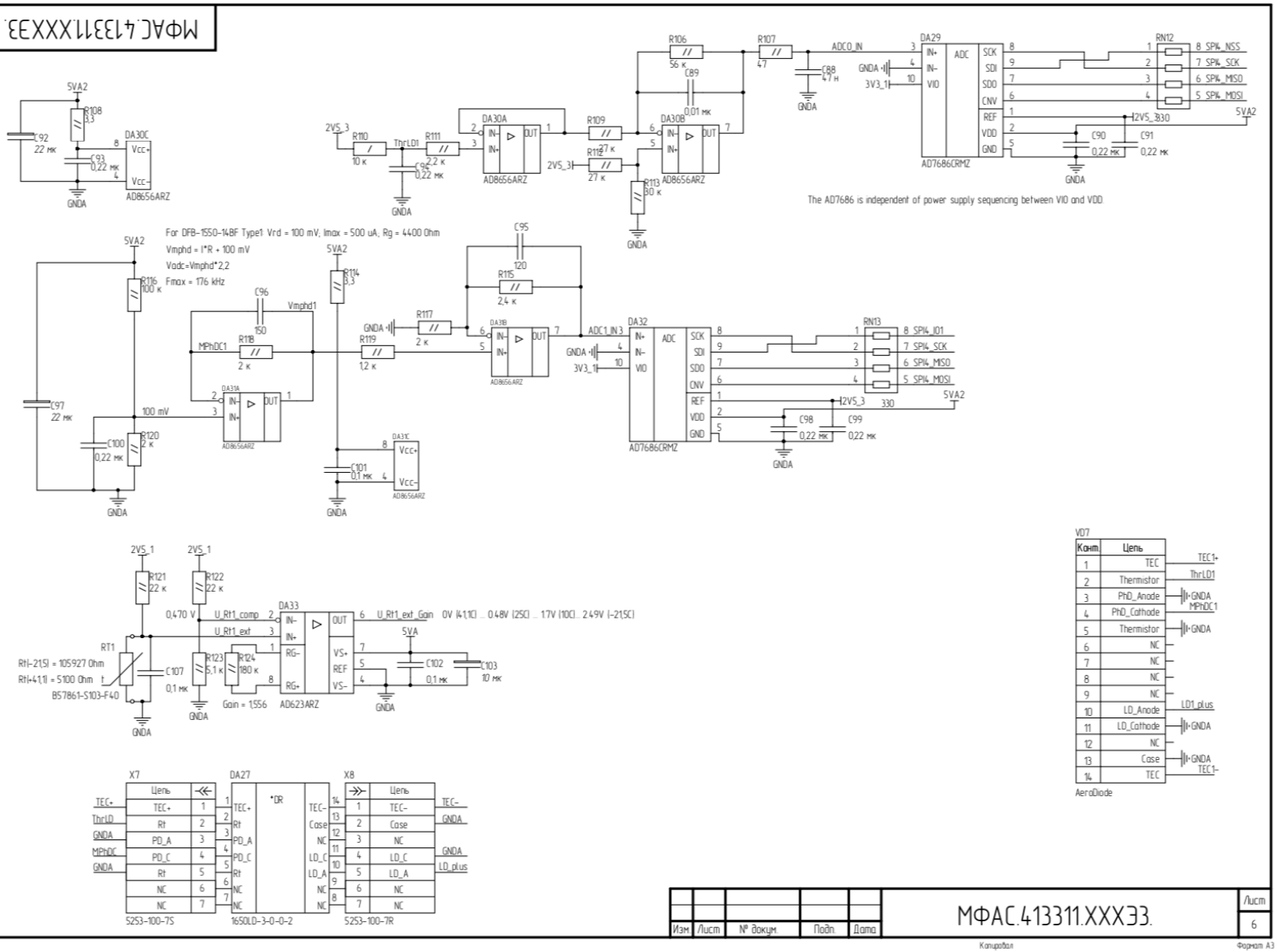
Изм	Лист	№ докум.	Подп.	Дата	МФАС.413311.XXX3.	Лист 5

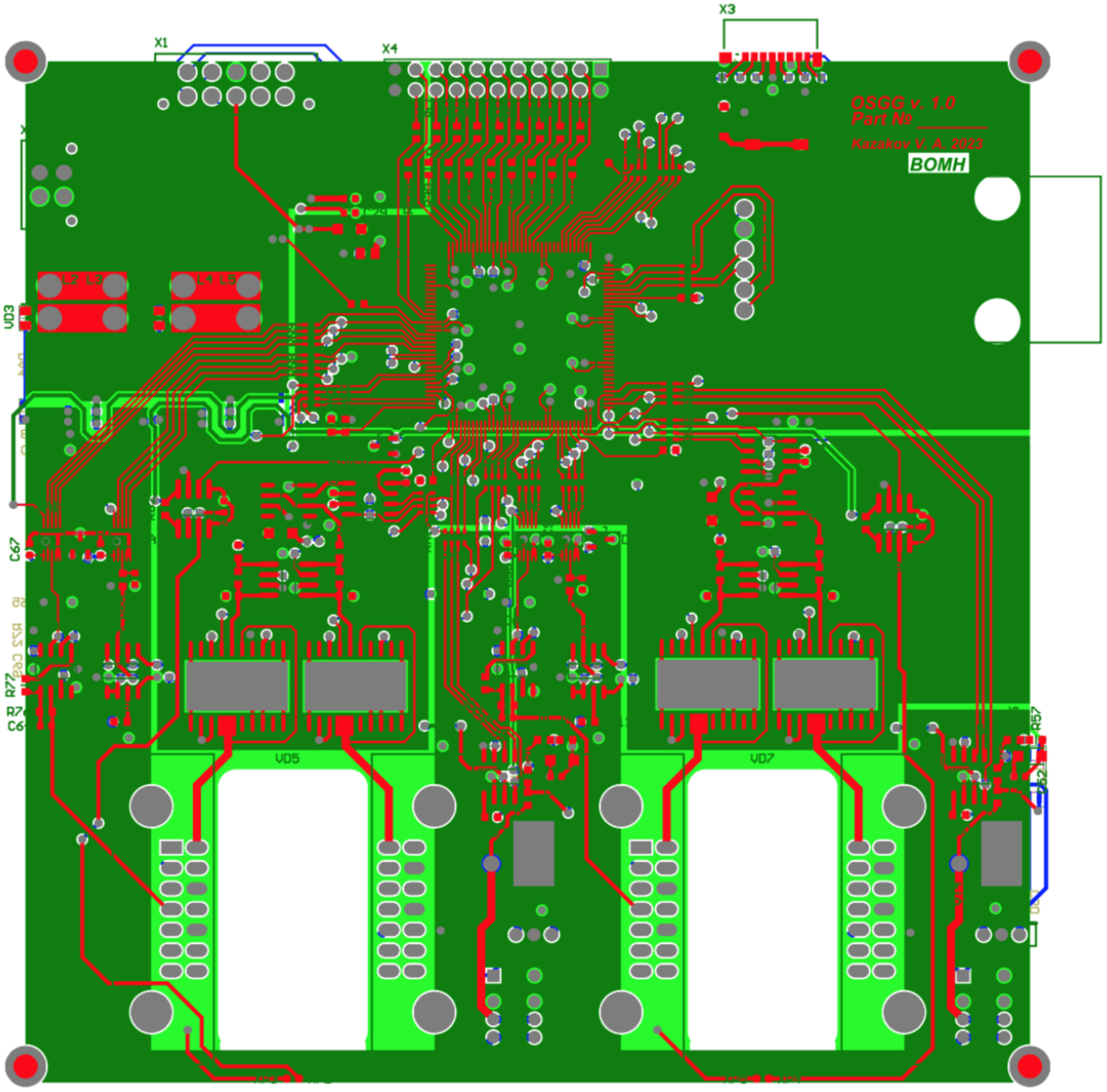
Копировал

Формат А3

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.


МФАС.413311.XXX33.
5VA2
R66 3.3
C60 22 мк
C61 0,22 мк
DA17C
AD8656ARZ
GNDA
2V5_2
R68 10 к
ThrLD2
R69 2,2 к
C62 0,22 мк
DA17A
AD8656ARZ
R64 56 к
R65 47
C57 0,01 мк
R67 27 к
R70 27 к
DA17B
AD8656ARZ
R71 30 к
ADC2_IN
C56 47 н
DA16
ADC
SCK
SDI
SDO
CNV
REF
VDD
GND
IN+
IN-
VIO
AD7686CRMZ
GNDA
3V3_1
RN8
8 SPI5_IO1
7 SPI5_SCK
6 SPI5_MISO
5 SPI5_MOSI
330
C58 0,22 мк
C59 0,22 мк
The AD7686 is independent of power supply sequencing between VIO and VDD.
For DFB-1550-14BF Type1 Vrd = 100 mV, Imax = 500 uA, Rg = 4400 Ohm
Vmphd = I*R + 100 mV
Vadc=Vmphd*2,2
Fmax = 176 kHz
R74 100 к
C64 150
Vmphd2
MPhDC2
R76 2 к
R77 1,2 к
R75 2 к
C63 120
R73 2,4 к
DA18A
DA18B
DA18C
C65 22 мк
100 mV
C68 0,22 мк
R78 2 к
C69 0,1 мк
ADC3_IN3
DA19
C66 0,22 мк
C67 0,22 мк
RN9
8 SPI5_NSS
7 SPI5_SCK
6 SPI5_MISO
5 SPI5_MOSI
2V5_1
R79 22 к
R80 22 к
0,470 V
U_Rt2_comp
U_Rt2_ext
U_Rt2_ext_Gain 0V (-41,1C) ... 0,48V (25C) ... 1,7V (10C)... 2,49V (-21,5C)
DA20
RT2
Rt(-21,5) = 105927 Ohm
Rt(+41,1) = 5100 Ohm
B57861-S103-F40
C106 0,1 мк
R81 5,1 к
R82 180 к
Gain = 1,556
AD623ARZ
5VA
C70 0,1 мк
C71 10 мк
VD5
DFB-1550-14BF Type1
TEC2+
ThrLD2
PhD_Anode
PhD_Cathode
LD2_plus
LD_Anode
LD_Cathode
TEC2-
X9
5253-100-7S
DA28
DFB-1550-14BF Type1
X10
5253-100-7R
TEC+
ThrLD
MPhDC
TEC-
Case
LD_C
LD_A
LD_plus
Изм.
Лист
№ докум.
Подп.
Дата
МФАС.413311.XXX33.
Лист
6
Копировал
Формат А3
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.


МФАС.413311.XXX33.
5VA2
R108 3.3
C92 22 мк
C93 0,22 мк
DA30C
Vcc+
Vcc-
AD8656ARZ
GNDA
2V5_3
R110 10 к
ThrLD1
R111 2,2 к
C94 0,22 мк
DA30A
DA30B
R109
R112 27 к
R113 30 к
R106 56 к
R107 47
C89 0,01 мк
C88 47 н
ADC0_IN
3V3_1
DA29
ADC
IN+
IN-
VIO
SCK
SDI
SDO
CNV
REF
VDD
GND
AD7686CRMZ
RN12
8_SPI4_NSS
7_SPI4_SCK
6_SPI4_MISO
5_SPI4_MOSI
330
12V5_3
C90 0,22 мк
C91 0,22 мк
The AD7686 is independent of power supply sequencing between VIO and VDD.
For DFB-1550-14BF Type1 Vrd = 100 mV, Imax = 500 uA, Rg = 4400 Ohm
Vmphd = I*R + 100 mV
Vadc=Vmphd*2,2
Fmax = 176 kHz
R116 100 к
C96 150
Vmphd1
MPhDC1
R118 2 к
R119 1,2 к
R117 2 к
C95 120
R115 2,4 к
R114 3.3
C97 22 мк
100 mV
C100 0,22 мк
R120 2 к
C101 0,1 мк
DA31A
DA31B
DA31C
ADC1_IN3
DA32
RN13
C98 0,22 мк
C99 0,22 мк
2V5_1
R121 22 к
R122 22 к
0,470 V
U_Rt1_comp
U_Rt1_ext
U_Rt1_ext_Gain
0V (-41,1C) ... 0,48V (25C) ... 1,7V (10C) ... 2,49V (-215C)
DA33
RG-
RG+
VS+
VS-
REF
5VA
RT1
Rt(-21,5) = 105927 Ohm
Rt(+41,1) = 5100 Ohm
BST861-S103-F40
C107 0,1 мк
R123 5,1 к
R124 180 к
Gain = 1,556
AD623ARZ
C102 0,1 мк
C103 10 мк
VD7
Conn
Цель
TEC
Thermistor
PhD_Anode
PhD_Cathode
LD_Anode
LD_Cathode
Case
NC
TEC1+
ThrLD1
LD1_plus
TEC1-
AeroDiode
X7
DA27
X8
TEC+
ThrLD
MPhDC
Rt
PD_A
PD_C
LD_C
LD_A
*DR
5253-100-7S
16501D-3-0-0-2
5253-100-7R
Изм.
Лист
№ докум.
Подп.
Дата
МФАС.413311.XXX33.
Лист
6
Копировал
Формат А3
Инв. № подл.
Подп. и дата
Взам. инв. №
Инв. № дубл.
Подп. и дата


X1
X4
X3
OSGG v. 1.0
Part №
Kazakov V. A. 2023
ВОМН
L2 L3
L4 L5
VD3
C67
R77
VD5
VD7
R87

Comment	Description	Designator	Footprint	LibRef	Quantity
0,1 мк	Chip-capacitor	C1, C2, C3, C5, C6, C7, C9, C10, C11, C12, C13, C14, C16, C18, C20, C22, C25, C26, C30, C31, C34, C40, C41, C44, C46, C48, C51, C52, C54, C69, C70, C72, C73, C76, C78, C80, C83, C85, C86, C101, C102, C106, C107	[0603]-C	Cap	43
1 мк	Chip-capacitor	C4, C19, C21	[0603]-C	Cap	3
4,7 мк	Tantal capacity	C8, C27	Type_A	Cap_tantal	2
2,2 мк	Chip-capacitor	C15, C17	[0603]-C	Cap	2
15	Chip-capacitor	C23, C24	[0603]-C	Cap	2
10 мк	Tantal capacity	C28, C33, C42, C43, C47, C55, C71, C74, C75, C79, C87, C103	Type_A	Cap_tantal	12
0,01 мк	Chip-capacitor	C29, C57, C89	[0603]-C	Cap	3
0,22 мк	Chip-capacitor	C32, C35, C36, C37, C38, C39, C45, C49, C58, C59, C61, C62, C66, C67, C68, C77, C81, C90, C91, C92, C94, C98, C99, C100, C104, C105	[0603]-C	Cap	26
10	Chip-capacitor	C50, C82	[0603]-C	Cap	2
22 мк	Tantal capacity	C53, C60, C65, C84, C93, C97	Type_A	Cap_tantal	6
47 н	Chip-capacitor	C56, C88	[0603]-C	Cap	2
120	Chip-capacitor	C63, C95	[0603]-C	Cap	2
150	Chip-capacitor	C64, C96	[0603]-C	Cap	2
MIC29303AWU	Стабилизатор напряжения регулируемый положительной полярности	DA1, DA4	d2pak5	MIC29303AWU	2
3.3V		DA2	SOT23-5	[illegible]	1
REF3125	Источник опорного напряжения фиксированный положительной полярности 2,5 В	DA3, DA5, DA6, DA7, DA34	SOT-23	REF3125	5
REF2041	Источник опорного напряжения фиксированный положительной полярности 2,048 В; 1,024 В	DA8, DA21	SOT-23-5	REF2041	2
OPA569		DA9, DA10, DA22, DA23	DWP20	OPA569	4
THS4531AID	Ultra Low-Power, Rail-to-Rail Output, Fully Differential Amplifier	DA11, DA24	SOIC8_2	THS4531AID	2
AD8571ARZ	Zero-Drift, Single-Supply, Rail-to-Rail Input/Output Operational Amplifiers	DA12, DA25	SOIC8	AD8571ARZ	2
AD5541ARZ	DAC 16 bit serial	DA13, DA15, DA26, DA28	SOIC8	AD5541ARZ	4
AD8656ARZ		DA14, DA17, DA18, DA27, DA30, DA31	SOIC8	AD8656ARZ	6
AD7686CRMZ	Serial ADC	DA16, DA19, DA29, DA32	RM-10_L	AD7686CRMZ	4
AD623ARZ	Single and Dual-Supply, Rail-to-Rail, Low Cost Instrumentation Amplifier	DA20, DA33	SOIC8_2	AD623ARZ	2
STM32F765ZET6	Cortex-M7 MCU by STM	DD1	LQFP-144	STM32F765ZET6	1
CP2102Board	USB to UART board	DD2	CP2102_board	CP2102_board	1
IMO3TS	Сигнальное реле	K1, K2	IM_THT	IMO3TS	2
BLM18PG331SN1	Filter EMI	L1, L2, L3, L4, L5	[0603]-BLM	BLM18PG331SN1	5
330		R1	[0603]-R	Res_0.125W	1
10 к		R2, R3, R4, R5, R6, R7, R8, R9, R12, R15, R20, R21, R32, R37, R46, R51, R53, R58, R88, R93, R95, R100, R129	[0603]-R	Res_0.125W	23
300		R10, R11, R13, R14, R16, R18, R19, R22, R23, R25, R26, R27, R28, R29, R30, R31, R125, R126	[0603]-R	Res_0.125W	18
100 к		R17, R74, R116	[0603]-R	Res_0.125W	3
0		R24	[0603]-R	Res_0.125W	1
300		R33, R38	[0603]-R	Res_0.125W	2
120		R34, R39	[0603]-R	Res_0.125W	2
47		R35	[0603]-R	Res_0.125W	1
100		R36, R40	[0603]-R	Res_0.125W	2
47		R41, R65, R83, R107	[0603]-R	Res_0.125W	4
1,5 к		R42, R48, R84, R90	[0603]-R	Res_0.125W	4
2 к		R43, R47, R75, R76, R78, R85, R89, R117, R118, R120, R131, R132	[0603]-R	Res_0.125W	12
9,1 к		R44, R45, R86, R87	[0603]-R	Res_0.125W	4
22 к		R49, R50, R79, R80, R91, R92, R121, R122	[0603]-R	Res_0.125W	8
68 к		R52, R94	[0603]-R	Res_0.125W	2
15 к		R54, R59, R96, R101	[0603]-R	Res_0.125W	4
7,5 к		R55, R60, R63, R97, R102, R105	[0603]-R	Res_0.125W	6
12 к		R56, R98	[0603]-R	Res_0.125W	2
20 к		R57, R99	[0603]-R	Res_0.125W	2
680		R61, R103	[0603]-R	Res_0.125W	2
30		R62, R104	[2512]RESC[illegible]	Res_1W	2
56 к		R64, R106	[0603]-R	Res_0.125W	2
3,3		R66, R72, R108, R114	[0603]-R	Res_0.125W	4
27 к		R67, R70, R109, R112	[0603]-R	Res_0.125W	4
10 к		R68, R110	[1206]RESC3216L	Res_0.25W	2
2,2 к		R69, R111	[0603]-R	Res_0.125W	2
30 к		R71, R113	[0603]-R	Res_0.125W	2
2,4 к		R73, R115	[0603]-R	Res_0.125W	2
1,2 к		R77, R119	[0603]-R	Res_0.125W	2
5,1 к		R81, R123	[0603]-R	Res_0.125W	2
180 к		R82, R124	[0603]-R	Res_0.125W	2
1 к		R127, R128, R130, R133, R134	[0603]-R	Res_0.125W	5
330		RN1, RN2, RN3, RN4, RN5, RN6, RN7, RN8, RN9, RN10, RN11, RN12, RN13	CAY-16	RES_SB	13
CPC1008N	Оптопара	U1, U2	SOP4	CPC1008N	2
KP-2012SGC	Светодиод зеленый 2x1,2x1,1 15мКд	VD1, VD2, VD3	[0805]-VD	KP-2012SGC	3
MBR130	Предохранительный диод	VD4, VD6	SOD-123	MBR130	2
DFB-1550-14BF Type1	Butterfly laser diode in China connector	VD5	Butterfly_CHINA	AeroDiode	1
AeroDiode	Butterfly laser diode in China connector	VD7	Butterfly_CHINA	AeroDiode	1
BC817	Биполярный p-n-p транзистор	VT1, VT2, VT5, VT6	SOT-23	BC817	4
PC817	Транзисторная оптопара	VT3, VT7	SOIC4	PC817	2
2SC3807	Биполярный n-p-n транзистор	VT4, VT8	TO-126	2SC3807	2
Micro-Fit_2x5		X1	Micro-Fit_2x5	Micro-Fit_2x5	1
Micro-Fit_2x2		X2	Micro-Fit_2x2	Micro-Fit_2x2	1
104031-0811	Card socket	X3	104031-0811	104031-0811	1
PLD-32		X4	PLD-32	PLD-32	1
PIN	PIN	XP1, XP2, XP3, XP4, XP5	LAM_PIN1	PIN	5
25MHz	Standart quartz resonator	ZQ1	HC49-SMD	Quartz	1

Приложение 1.1.2.2 Параметры протокола обмена данными между Платой электронной управляющей и Персональным компьютером

Плата электронная управляющая принимает команды и передает данные в персональный компьютер (ПК). Обмен данными между управляющей платой и ПК происходит по отдельному каналу связи. Интерфейс связи - последовательный асинхронный, который логически совместим с интерфейсом UART. Формат передачи байта: старт-бит, 8 бит данных, без бита четности, стоп-бит.

Скорость передачи данных – 115200 бит/сек. Режим работы интерфейса платы – полнодуплексный.

Команды.

Для управления драйвера лазеров предусмотрено 6 команд: 1 команда длинного формата (426 байт или 213 16битных слов) и 5 команд короткого формата (2 байта или одно 16битное слово). Каждая команда начинается с заголовка. Он определяет тип команды. На управляющую плату передаются команды со значениями заголовка: 0x1111, 0x2222, 0x3333, 0x4444, 0x5555 и 0x6666. В случае короткого формата, команда состоит только из заголовка. Все команды имеют 16-битный формат. При этом первым байтом каждого слова является младший байт, а вторым – старший.

Если при посылки команды указана неправильная метка, прибор выключает всю периферию и посылает ОС 1 (см. ниже).

УКС1. Команда настроек (DECODE_ENABLE).

Формат командного слова представлен в Табл. 1.

Табл. 1. Формат команды УКС1.

№ слова	Название	Код	Расшифровка
0	Header	0x1111	Заголовок
1	Setup	0xXXXX	Биты настроек:

			0 – Work Enable 1 – 5V1 Enable 2 – 5V2 Enable 3 – LD1 Enable 4 – LD2 Enable 5 – Ref1 Enable 6 – Ref2 Enable 7 – TEC1 Enable 8 – TEC2 Enable 9 – TS1 Enable 10 – TS2 Enable 11 – SD Enable 12 – PI1 RD 13 – PI2 RD 14 – Reserved 15 – Reserved
2	LD1_temp	0xXXXX	Температура ЛМ1
3	LD2_temp	0xXXXX	Температура ЛМ2
4	Reserved	0x0000	Зарезервировано
5	Reserved	0x0000	Зарезервировано
6	Reserved	0x0000	Зарезервировано
7	P_temp1	0xXXXX	Пропорциональный коэффициент для ЛМ1
8	I_temp1	0xXXXX	Интегральный коэффициент для ЛМ1
9	P_temp2	0xXXXX	Пропорциональный коэффициент для ЛМ2
10	I_temp2	0xXXXX	Интегральный коэффициент для ЛМ2
11	Message_ID	0xXXXX	Номер сообщения
12	I_1_0	0xXXXX	Табличные значения тока ЛМ1
13	I_1_1	0xXXXX	
…	…	…	
111	I_1_100	0xXXXX	

112	I_2_0	0xXXXX	Табличные значения тока ЛМ2
113	I_2_1	0xXXXX	
…	…	…	
211	I_2_100	0xXXXX	
212	CRC	0xXXXX	Контрольная сумма

Данная команда подаётся для задания основного режима работы драйвера лазеров.

После заголовка идёт слово, состоящее из битов настроек:

0 – Work Enable – разрешает работу драйвера лазеров;

1 – 5V1 Enable – включает напряжение питания драйверов Пельтье и внешних датчиков температуры;

2 – 5V2 Enable – включает напряжение питания драйверов тока лазера, внутренних датчиков температуры, усилителей мониторного канала;

3 – LD1 Enable – разрешает работу драйвера тока первого лазерного модуля (ЛМ1);

4 – LD2 Enable – разрешает работу драйвера тока второго лазерного модуля (ЛМ2);

5 – Ref1 Enable – включает ИОН драйвера Пельтье ЛМ1;

6 – Ref2 Enable – включает ИОН драйвера Пельтье ЛМ2;

7 – TEC1 Enable – включает выходной усилитель драйвера Пельтье ЛМ1;

8 – TEC2 Enable – включает выходной усилитель драйвера Пельтье ЛМ2;

9 – TS1 Enable – разрешает работу процедуры термостабилизации ЛМ1;

10 – TS2 Enable – разрешает работу процедуры термостабилизации ЛМ2;

11 – SD Enable – разрешает запись данных в память microSD;

12 – PI1 RD – разрешает чтение параметров ПИ-регулятора из команды для ЛМ1;

13 – PI2 RD – разрешает чтение параметров ПИ-регулятора из команды для ЛМ2;

14 – Reserved – зарезервировано;

15 – Reserved – зарезервировано.

Температуры LD1_temp и LD2_temp в теле команды задаются в значениях АЦП ($N_{2,3}$), к которым должна стремиться система термостабилизации. Для пересчёта значений температуры (T, ºC) в значения АЦП ($N_{2,3}$) необходимо воспользоваться следующими формулами:

$$R_t = 10000 \cdot \exp exp\left(\frac{3900}{T+273} - \frac{3900}{298}\right), \quad (1)$$

$$U = \frac{V_{REF}}{R_5(R_3+R_4)} \bullet \frac{R_1R_4(R_5+R_6) - R_t(R_3R_6 - R_4R_5)}{R_t+R_1}, \quad (2)$$

$$N_{2,3} = U\frac{65535}{V_{REF}}, \quad (3)$$

где R_t – сопротивление термистора [Ом], U – напряжение на термисторе [В],

V_{REF} = 2,5 В – опорное напряжение,

R_1 = 10000 Ом,

R_2 = 2200 Ом,

R_3 = 27000 Ом,

R_4 = 30000 Ом,

R_5 = 27000 Ом,

R_6 = 56000 Ом (см. Рис. 1).

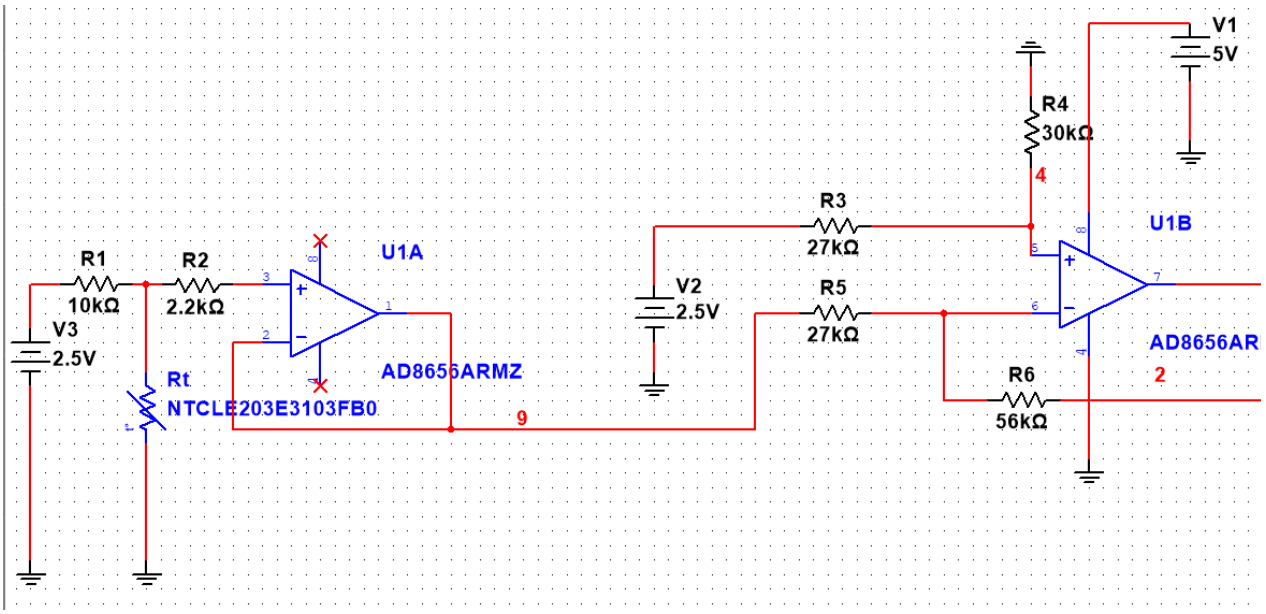


Рис. 1. Схема усилителя термистора.

Слова 4, 5, 6 являются зарезервированными для случая модернизации и для текущей версии прошивки должны быть равны нулю.

Слова 7, 8 – коэффициенты для ПИ-регулятора стабилизатора температуры ЛМ1, а 9, 10 – ЛМ2. Коэффициенты каждого лазера подбираются отдельно.

Слово 11 – номер команды.

Слова с 12 по 111 – табличные значения тока ЛМ1, выраженные в числах ЦАПа ($N_{12...111}$).

Слова со 112 по 211 – табличные значения тока ЛМ2, выраженные в числах ЦАПа ($N_{112...211}$).

Для перевода значений тока лазеров (I_{LM}) в числа ЦАПа ($N_{12...211}$), необходимо воспользоваться следующей формулой:

$$N_{12...211} = \frac{65535}{2000} R_{REF} I_{LM}[\text{мА}], \quad (4)$$

где R_{REF} = 30 Ом – значение токозадающего резистора [Ом].

Каждый бит контрольной суммы CRC (последнее слово) является суммой по модулю 2 соответствующих бит каждого слова за исключением заголовка.

При несовпадении контрольной суммы, неправильного заголовка, а также при всевозможных других ошибках передачи команды, плата игнорирует принятую команду, установит бит «ошибка команды» в ответных данных (UART_ERR в ОС1, см. ниже) и продолжит работу в соответствии с последней принятой достоверной командой и собственной циклограммой работы. Плата отвечает на данную команду посылкой ОС 1.

УКС 2. Возврат к настройкам по умолчанию (DEFAULT_ENABLE).

УКС 2 является двухбайтовой командой со значением 0x2222. Команда возвращает устройство в состояние по умолчанию: все настройки сброшены, вся периферия выключена.

Плата отвечает на данную команду посылкой ОС 1.

УКС 3. Передать сохранённые данные (TRANSS_ENABLE).

УКС 3 является двухбайтовой командой со значением 0x3333. По команде передаются данные, сохранённые на флэш память. Перед началом скачивания сохранённых данных драйвер должен быть сброшен (УКС 2).

Плата отвечает на данную команду посылкой ОС 3.

УКС 4. Передать текущие данные (TRANS_ENABLE).

УКС 4 является двухбайтовой командой со значением 0x4444. По команде передаются последние сформированные данные.

Плата отвечает на данную команду посылкой ОС 2.

УКС 5. Удалить сохранённые данные (REMOVE_FILE).

УКС 5 является двухбайтовой командой со значением 0x5555. По команде удаляется файл, содержащий на флэш памяти сохранённые данные.

Плата отвечает на данную команду посылкой ОС 1.

УКС 6. Запрос статуса (STATE).

УКС 6 является двухбайтовой командой со значением 0x6666. Ответом команды является посылка, информирующая о состоянии устройства.

Плата отвечает на данную команду посылкой ОС 1.

Данные

Всего предусмотрено 3 типа посылок, содержащие данные.

ОС 1. Статус (STATE).

ОС 1 является двухбайтовой посылкой, каждый бит которой информирует об ошибках в работе драйвера (см. Табл. 2).

Табл. 2. Формат ОС 1.

№ бита	Название	Маска/значение	Расшифровка
0	SD_ERR	0x0001	Ошибка чтения/записи SD-карты
1	UART_ERR	0x0002	Ошибка команды (неправильная метка или

			превышен таймаут передачи команды (1 секунда))
2	UART_DECODE_ERR	0x0004	Неверно заданы параметры
3	TEC1_ERR	0x0008	Перегрев драйвера ТЕС у ЛМ1
4	TEC2_ERR	0x0010	Перегрев драйвера ТЕС у ЛМ2
5	DEFAULT_ERR	0x0020	Ошибка сброса системы
6	REMOVE_ERR	0x0040	Ошибка удаления файла
7	Reserved	0x0080	Зарезервировано
8	Reserved	0x0100	Зарезервировано
9	Reserved	0x0200	Зарезервировано
10	Reserved	0x0400	Зарезервировано
11	Reserved	0x0800	Зарезервировано
12	Reserved	0x1000	Зарезервировано
13	Reserved	0x2000	Зарезервировано
14	Reserved	0x4000	Зарезервировано
15	Reserved	0x8000	Зарезервировано

Посылка приходит в ответ на команды: УКС 1, УКС 2, УКС 5 и УКС 6. Устройство, обнаружив неправильную метку немедленно перейдёт в первоначальный режим, отключив всю периферию и выдаст код ошибки 0x0002.

Каждая команда должна быть передана в течение 1 секунды. В противном случае выдаётся ОС 1 = 0x0002 и сбрасывается счётчик команд. При этом прибор продолжит работу по последней принятой команде настроек. Сделано это для отлавливания сбоев, при которых в команде выпадает старший или младший байт слова или же не присылается неполная команда. В случае получения ошибки 0x0002, необходимо отправить последнюю команду ещё раз с самого начала. При повторном получении ошибки необходимо проверить метку команды.

ОС 2. Пакет данных (DATA).

ОС 2 являются пакетом, содержащим последние сформированные в ходе работы драйвера данными.

Формат ОС 2 представлен в Табл. 3.

Табл. 3. Формат ОС 2.

№ слова	Название	Код	Расшифровка
0	Header	0x1111	Заголовок
1	P_1_0	0xXXXX	Фототок
2	P_1_1	0xXXXX	мониторного
…	…	…	фотодиода ЛМ1
100	P_1_100	0xXXXX	
101	P_2_0	0xXXXX	Фототок
102	P_2_1	0xXXXX	мониторного
…	…	…	фотодиода ЛМ2
200	P_2_100	0xXXXX	
201	TO_LSB	0xXXXX	Основной таймер
202	TO_MSB	0xXXXX	
203	T_LM1	0xXXXX	Температура ЛМ1
204	T_LM2	0xXXXX	Температура ЛМ2
205	T_EXT1	0xXXXX	Температура первого внешнего термистора
206	T_EXT2	0xXXXX	Температура второго внешнего термистора
207	3V3MON	0xXXXX	Напряжение 3,3 В
208	5V1MON	0xXXXX	Напряжение питания драйверов Пельтье и внешних датчиков температуры
209	5V2MON	0xXXXX	Напряжение питания драйверов тока лазера, внутренних датчиков температуры, усилителей мониторного канала
210	7V0MON	0xXXXX	Входное напряжение питания

211	Message_ID	0xXXXX	Номер сообщения
212	CRC	0xXXXX	Контрольная сумма

После заголовка идут слова с 1 по 100 – это значения фототока ЛМ1, выраженные в числах АЦП (N_{ADC1}).

Слова со 101 по 200 –значения тока ЛМ2, выраженные в числах АЦП (N_{ADC2}).

Пересчёт значений АЦП в значения фототока производится по формуле (5):

$$I_{PhD} = \frac{N_{1...200} \cdot 2{,}5}{65535 \cdot 4{,}4} - \frac{1}{20{,}4}\ [\text{мА}], \quad (5)$$

Слова 201 и 202 – основной таймер драйвера, начинающий отсчёт после включения питания. Таймер сбрасывается после подачи команды УКС 2. Отсчитываются каждые 10 мс.

Слова 203 и 204 – текущие температуры ЛМ1 и ЛМ2 соответственно, выраженной в числах АЦП. Формулы для перевода значений АЦП в значения температуры приведены ниже:

$$U = \frac{N_{203,\ 204} V_{REF}}{65535}\ [\text{В}], \quad (6)$$

$$R_t = \frac{R_1\left(V_{REF} R_4\left(R_5+R_6\right) - U R_5\left(R_3+R_4\right)\right)}{U R_5\left(R_3+R_4\right) + V_{REF} R_3 R_6 - V_{REF} R_4 R_5}\ [\text{Ом}], \quad (7)$$

$$T_{LM} = \frac{1}{\frac{1}{298} + \frac{1}{3900} \ln ln\left(\frac{R_t}{10000}\right)} - 273\ [°C], \quad (8)$$

Значения параметров в формулах (6), (7) и (8) равны значениям параметров в формулах (1), (2) и (3).

Слова 205 и 206 – температуры внешних термисторов, выраженных в числах АЦП. Формулы для перевода значений АЦП в значения температуры приведены ниже:

$$U = \frac{N_{205,\ 206} V_{REF}}{4095} \frac{1}{1+\frac{100000}{R_{10}}} + V_{REF} \frac{R_9}{R_8+R_9}\ [\text{В}], \quad (9)$$

$$R_t = R_7 \frac{U}{E-U}\ [\text{Ом}], \quad (10)$$

$$T_{LM} = \frac{1}{\frac{1}{298} + \frac{1}{3455} \ln ln\left(\frac{R_t}{10000}\right)} - 273\ [°C], \quad (11)$$

где $R_7 = R_8 = 22000$ Ом; $R_9 = 5100$ Ом; $V_{REF} = 2{,}5$ В (см. Рис. 2).

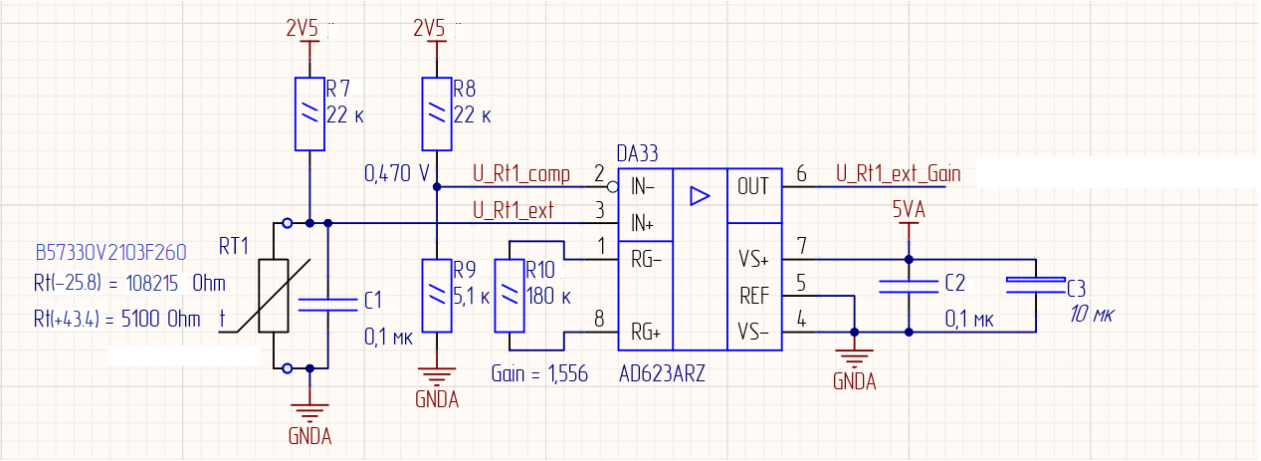


Рис. 2. Схема усилителя внешнего термистора.

Далее идут оцифрованные значения питающих напряжений, которые пересчитываются по следующим формулам:

$$U_{3V3} = N_{207} \cdot 1,221 \cdot 10^{-3} [\text{В}], \quad (12)$$

$$U_{5V1,\ 5V2} = N_{208,\ 209} \cdot 1,9315 \cdot 10^{-3} [\text{В}], \quad (13)$$

$$U_{7V0} = N_{210} \cdot 6,72 \cdot 10^{-3} [\text{В}], \quad (14)$$

Слово 211 содержит номер последней принятой команды.

Аналогично с УКС 1 каждый бит контрольной суммы CRC (слово 212) является суммой по модулю 2 соответствующих бит каждого слова за исключением заголовка.

ОС 3. Пакет сохранённых данных (SAVED DATA).

Формат посылок отличается от ОС 2 только первым пакетом, который содержит в первых 2 словах размер передаваемой информации, выраженной в **байтах**.

Организация сеансов связи с драйвером.

После включения питания может быть подана любая команда.

Рекомендуется перед сменой режима работы драйвера подавать команду УКС 2. Интервал следования команд должен быть не менее 100 мс.

Перед выключением питания необходимо отправить команду сброса (УКС 2).

Примеры сеанса связи.

Команда	Ответ
Включение питания	
УКС1 Команда настроек	ОС1 Статус
УКС4 Передать текущие данные	ОС2 Пакет данных
…	…
УКС4 Передать текущие данные	ОС2 Пакет данных
УКС2 Возврат к настройкам по умолчанию	ОС1 Статус
УКС1 Команда настроек	ОС1 Статус
УКС4 Передать текущие данные	ОС2 Пакет данных
…	…
УКС4 Передать текущие данные	ОС2 Пакет данных
УКС2 Возврат к настройкам по умолчанию	ОС1 Статус
УКС3 Передать сохранённые данные	ОС3 Пакет сохранённых данных
…	…
УКС3 Передать сохранённые данные	ОС3 Пакет сохранённых данных
УКС2 Возврат к настройкам по умолчанию	ОС1 Статус
Выключение питания	

Приложение 1.1.2.3 Код программного обеспечения управления лазерными диодами

device_main.py

```
from PySimpleGUI import TIMEOUT_KEY, WIN_CLOSED

import device_interaction as dev
from  device_commands import I_POINTS_SET
import gui

#### ---- Constants

GUI_TIMEOUT_INTERVAL = 505 - dev.WAIT_AFTER_SEND # GUI refresh time in milliseconds

SAVE_POINTS_NUMBER = 1000 # Number of most recent data points kept in memory

INITIAL_TEMPERATURE_1 = 22.40 # Set initial temperature for Laser 1 in Celsius: from -1 to 45 C ??
INITIAL_TEMPERATURE_2 = 16.70 # Set initial temperature for Laser 2 in Celsius: from -1 to 45 C ??
INITIAL_CURRENT_1 = 32.0 # 64.0879 max # Set initial current for Laser 1, in mA
INITIAL_CURRENT_2 = 32.0 # 64.0879 max # Set initial current for Laser 2, in mA

#### ---- Functions

def shorten(i):
    return "{:.2f}".format(round(i, 2))



def set_initial_params():
    params = {}
    params['Temp_1'] = INITIAL_TEMPERATURE_1 # Initial temperature for Laser 1
    params['Temp_2'] = INITIAL_TEMPERATURE_2 # Initial temperature for Laser 2
    params['ProportionalCoeff_1'] = int(10*256) # Proportional coefficient for temperature stabilizatoin for
Laser 1
    params['ProportionalCoeff_2'] = int(10*256) # Proportional coefficient for temperature stabilizatoin for
Laser 2
    params['IntegralCoeff_1'] = int(0.5*256) # Integral coefficient for temperature stabilizatoin for Laser 1
    params['IntegralCoeff_2'] = int(0.5*256) # Integral coefficient for temperature stabilizatoin for Laser 2
    params['Message_ID'] = "00FF" # Send Message ID (hex format)
    params['Iset_1'] = [INITIAL_CURRENT_1 for i in range(I_POINTS_SET)] # Currency value array for Laser 1,
in mA
```

```
    params['Iset_2'] = [INITIAL_CURRENT_2 for i in range(I_POINTS_SET)] # Currency value array for Laser 2,
in mA
    return params


def update_data_lists():
    saved_data.append(data)
    if len(saved_data)>SAVE_POINTS_NUMBER:
        saved_data.pop(0)

    draw_data.append(data)
    if len(draw_data)>gui.GRAPH_POINTS_NUMBER:
        draw_data.pop(0)


######## ---- Main program

prt=dev.setup_connection()

dev.reset_port_settings(prt)
# dev.request_state(prt)

params = set_initial_params()
dev.send_control_parameters(prt, params)

saved_data = []
draw_data = []

saved_data.append(dev.request_data(prt))
draw_data.append(saved_data[0])

window = gui.setup_gui(params)
axes_signs = gui.sign_axes(window)

while True:
    event, values = window.read(timeout=GUI_TIMEOUT_INTERVAL)
    # print(event, values)
    if event == WIN_CLOSED or event == '-EXIT-':
        break

    elif event == '-SetT1-':
```

```
        temp = float(values['-InputT1-'])
        params['Temp_1'] = temp
        dev.send_control_parameters(prt, params)
    elif event == '-SetT2-':
        temp = float(values['-InputT2-'])
        params['Temp_2'] = temp
        dev.send_control_parameters(prt, params)
    elif event == '-SetI1-':
        curr = float(values['-InputI1-'])
        params['Iset_1'] = [curr for i in range(I_POINTS_SET)]
        dev.send_control_parameters(prt, params)
    elif event == '-SetI2-':
        curr = float(values['-InputI2-'])
        params['Iset_2'] = [curr for i in range(I_POINTS_SET)]
        dev.send_control_parameters(prt, params)


    elif event == TIMEOUT_KEY:
        data = dev.request_data(prt)

        # print(data)
        update_data_lists()

        window['-TOUT_1-'].update(gui.READ_TEMPERATURE_TEXT+' 1: '+shorten(data['Temp_1'])+' C')
        window['-TOUT_2-'].update(gui.READ_TEMPERATURE_TEXT+' 2: '+shorten(data['Temp_2'])+' C')
        window['-IOUT_1-'].update(gui.READ_CURRENT_TEXT+' 1: '+shorten(data['I1'][0])+' мА')
        window['-IOUT_2-'].update(gui.READ_CURRENT_TEXT+' 2: '+shorten(data['I2'][0])+' мА')
        window['-DateTime-'].update(data['datetime'].strftime('%d-%m-%Y %H:%M:%S:%f')[:-3])
        window['-TTerm1-'].update('T терм 1: '+shorten(data['Temp_Ext_1'])+' C')
        window['-TTerm2-'].update('T терм 2: '+shorten(data['Temp_Ext_2'])+' C')
        window['-3V3-'].update('3V3: '+shorten(data['MON_3V3'])+' B')
        window['-5V1-'].update('5V1: '+shorten(data['MON_5V1'])+' B')
        window['-5V2-'].update('5V2: '+shorten(data['MON_5V2'])+' B')
        window['-7V0-'].update('7V0: '+shorten(data['MON_7V0'])+' B')

        window['-GraphT1-'].draw_line((len(draw_data)-1, draw_data[-2]['Temp_1']), (len(draw_data),
draw_data[-1]['Temp_1']), color='yellow')
        window['-GraphT2-'].draw_line((len(draw_data)-1, draw_data[-2]['Temp_2']), (len(draw_data),
draw_data[-1]['Temp_2']), color='yellow')
        window['-GraphI1-'].draw_line((len(draw_data)-1, draw_data[-2]['I1'][-1]), (len(draw_data),
draw_data[-1]['I1'][-1]), color='yellow')
```

```
        window['-GraphI2-'].draw_line((len(draw_data)-1, draw_data[-2]['I2'][-1]), (len(draw_data),
draw_data[-1]['I2'][-1]), color='yellow')

        # When graphs reach end of X scale, start scrolling
        if len(draw_data)>=gui.GRAPH_POINTS_NUMBER:
            # Scroll graphs
            window['-GraphT1-'].move(-1, 0)
            window['-GraphT2-'].move(-1, 0)
            window['-GraphI1-'].move(-1, 0)
            window['-GraphI2-'].move(-1, 0)

            # Scroll back graphs' labels
            for key, sgn in axes_signs.items():
                window[key].MoveFigure(sgn[0], 1, 0)
                window[key].MoveFigure(sgn[1], 1, 0)
        # dev.print_data(data)
        # print(event, values)
window.close()
dev.close_connection(prt)
```

device_interaction.py

```
import time

from datetime import datetime
import device_commands as cmd

#### ---- Constants

WAIT_AFTER_SEND = 0.3 # Wait after sending command, before requesting input (in seconds).

#### ---- High-level port commands

def setup_connection():
    prt = cmd.setup_port_connection()
    cmd.open_port(prt)
    return prt



def reset_port_settings(prt):
    # Reset port settings and check status
```

```
    cmd.send_DEFAULT_ENABLE(prt)
    time.sleep(WAIT_AFTER_SEND)
    status = cmd.get_STATE(prt).hex()
    if status is not None:
        print("Received: STATE. State status:", cmd.decode_STATE(status), "("+cmd.flipfour(status)+")")
        print("")


def request_state(prt):
    # Request data
    cmd.send_STATE(prt)
    time.sleep(WAIT_AFTER_SEND)
    status = cmd.get_STATE(prt).hex()
    if status is not None:
        print("Received: STATE. State status:", cmd.decode_STATE(status), "("+cmd.flipfour(status)+")")
        print("")


def send_control_parameters(prt, params):
    # Send control parameters
    hexstring = cmd.encode_Input(params)
    cmd.send_DECODE_ENABLE(prt,hexstring)
    time.sleep(WAIT_AFTER_SEND)
    status = cmd.get_STATE(prt).hex()
    if status is not None:
        print("Received: STATE. State status:", cmd.decode_STATE(status), "("+cmd.flipfour(status)+")")
        print("")
    else:
        print("")


def request_data(prt):
    # Request data
    cmd.send_TRANS_ENABLE(prt)
    time.sleep(WAIT_AFTER_SEND)
    data = cmd.get_DATA(prt).hex()
    data_dict = []
    if data is not None:
        data_dict = cmd.decode_DATA(data)
    return data_dict
```

```
def print_data(data):
    def shorten(i):
        return str(round(i, 2))

    print("Data from device (time: "+datetime.now().strftime("%H:%M:%S:%f")+"):")
    print("Message Header:", data['Header'], "  Message ID:", data['Message_ID'])
    print("Photodiode Current 1 ("+str(len(data['I1']))+" values):", \
        shorten(data['I1'][0]), shorten(data['I1'][1]), "...", \
        shorten(data['I1'][-2]), shorten(data['I1'][-1]), "mA")
    print("Photodiode Current 2 ("+str(len(data['I2']))+" values):", \
        shorten(data['I2'][0]), shorten(data['I2'][1]), "...", \
        shorten(data['I2'][-2]), shorten(data['I2'][-1]), "mA")
    print("Laser Temperature 1:", shorten(data['Temp_1']), "C")
    print("Laser Temperature 2:", shorten(data['Temp_2']), "C")
    print("Temperature of external thermistor 1:", shorten(data['Temp_Ext_1']), "C")
    print("Temperature of external thermistor 2:", shorten(data['Temp_Ext_2']), "C")
    print("Voltages 3V3: "+shorten(data['MON_3V3'])+"V   5V1: "+shorten(data['MON_5V1'])+ \
        "V   5V2: "+shorten(data['MON_5V2'])+"V   7V0: "+shorten(data['MON_7V0'])+"V.")

def close_connection(prt):
    cmd.close_port(prt)
```

device_commands.py

```
import serial
import device_conversion as cnv
from datetime import datetime

#### ---- Constants

GET_DATA_TOTAL_LENGTH = 426 # Total number of bytes when getting DATA
SEND_PARAMS_TOTAL_LENGTH = 426 # Total number of bytes when sending parameters
I_POINTS_SET = 100 # Number of points when setting I
I_POINTS_GET = 100 # Number of points when getting I


#### ---- Auxiliary functions

def int_to_hex(inp):
    if inp<0 or inp>65535:
```

```
        print("Error. Input should be within [0, 65535]. Returning N=0.")
        return "0000"
    return f"{inp:#0{6}x}"[2:]

def crc(lst):
    crc=int("0x"+lst[0],16)
    for i in range(1,len(lst)):
        crc=crc^int("0x"+lst[i],16)
    return int_to_hex(crc)

def show_hex_string(string):
    return "".join("\\x{}".format(char.encode()) for char in (string[i:i+2] for i in range(0, len(string), 2)))



def flipfour(s):
    ''' Changes "abcd" to "cdba"
    '''
    if len(s) != 4:
        print("Error. Trying to flip string with length not equal to 4.")
        return None
    return s[2:4]+s[0:2]




#### ---- Port Operations

def setup_port_connection():
    prt = serial.Serial()
    prt.baudrate = 115200
    prt.port = 'COM3'
    prt.timeout = 1 # in seconds
    return prt


def open_port(prt):
    prt.open()

    if prt.is_open:
        print("Connection succesful. Port is opened.")
        print("Port parameters:", prt)
```

```
        print("")
    else:
        print("Can't open port. Exiting program.")
        exit()


def close_port(prt):
    prt.close()
    print("")
    if prt.is_open:
        print("Can't close port. Exiting program.")
        exit()
    else:
        print("Port is closed. Exiting program.")
        exit()


#### ---- Interacting with device: low-level

# ---- Sending commands

def send_DECODE_ENABLE(prt, bytestring):
    ''' Set control parameters (x1111 + ...).
        Expected device answer: STATE.
    '''
    if len(bytestring) != SEND_PARAMS_TOTAL_LENGTH:
        print("Error. Wrong parameter string for DECODE_ENABLE.")
        return None
    prt.write(bytestring)
    print("Sent: Set control parameters (DECODE_ENABLE).")


def send_DEFAULT_ENABLE(prt):
    ''' Reset the device (x2222).
        Expected device answer: STATE.
    '''
    input = bytearray.fromhex(flipfour("2222"))
    prt.write(input)
    print("Sent: Reset device (DEFAULT_ENABLE).")
```

```
def send_TRANSS_ENABLE(prt):
    ''' Request all saved data (x3333).
        Expected device answer: SAVED_DATA.
    '''
    #
    pass


def send_TRANS_ENABLE(prt):
    ''' Request last piece of data (x4444).
        Expected device answer: DATA.
    '''
    input = bytearray.fromhex(flipfour("4444"))
    prt.write(input)
    print("Sent: Request last data (TRANS_ENABLE).")


def send_REMOVE_FILE(prt):
    ''' Delete saved data (x5555).
        Expected device answer: STATE.
    '''
    input = bytearray.fromhex(flipfour("5555"))
    prt.write(input)
    print("Sent: Delete saved data (REMOVE_FILE).")
    pass


def send_STATE(prt):
    ''' Request state (x6666).
        Expected device answer: STATE.
    '''
    input = bytearray.fromhex(flipfour("6666"))
    prt.write(input)
    print("Sent: Request state (STATE).")
    pass


# ---- Getting data


def get_STATE(prt):
```

```
    ''' Get decoded state of the device in byte format (2 bytes).
    '''

    print("Received "+str(prt.inWaiting())+" bytes.")
    if prt.inWaiting()!=2:
        print("Error. Couldn't get STATE data. prt.inWaiting():", prt.inWaiting())
        print("Flushing input data:", prt.read(prt.inWaiting()))
        # print("Flushing input data:", prt.read(2), prt.read(2))
        return None

    out_bytes = prt.read(2)
    return out_bytes


def get_DATA(prt):
    ''' Get decoded state of the device in byte format (426 bytes).
    '''

    print("Received "+str(prt.inWaiting())+" bytes.\n")
    if prt.inWaiting()!=GET_DATA_TOTAL_LENGTH:
        print("Error. Couldn't get DATA data.")
        return None

    out_bytes = prt.read(GET_DATA_TOTAL_LENGTH)
    return out_bytes


#### ---- Interacting with device: decode/encode messages

# ---- Encoding functions

def encode_Setup():
    bits=['0']*16

    bits[15] = "1" # enable work
    bits[14] = "1" # enable 5v1
    bits[13] = "1" # enable 5v2
    bits[12] = "1" # enable LD1
    bits[11] = "1" # enable LD2
    bits[10] = "1" # enable REF1
    bits[9]  = "1" # enable REF2
```

```
    bits[8]  = "1" # enable TEC1
    bits[7]  = "1" # enable TEC2
    bits[6]  = "1" # enable temp stab 1
    bits[5]  = "1" # enable temp stab 2
    bits[4]  = "0" # enable sd save
    bits[3]  = "1" # enable PI1 coef read
    bits[2]  = "1" # enable PI2 coef read
    bits[1]  = "0" # reserved
    bits[0]  = "0" # reserved

    s="".join([str(i) for i in bits])
    return hex(int(s,2))[2:]


def encode_Input(params):

    if params is None:
        return bytearray.fromhex("1111"+"00"*424)

    data = flipfour("1111") # Word 0
    data += flipfour(encode_Setup()) # Word 1
    data += flipfour(int_to_hex(cnv.conv_T_C_to_N(params['Temp_1']))) # Word 2
    data += flipfour(int_to_hex(cnv.conv_T_C_to_N(params['Temp_2']))) # Word 3
    data += flipfour("0000")*3 # Words 4-6
    data += flipfour(int_to_hex(params['ProportionalCoeff_1'])) # Word 7
    data += flipfour(int_to_hex(params['IntegralCoeff_1'])) # Word 8
    data += flipfour(int_to_hex(params['ProportionalCoeff_2'])) # Word 9
    data += flipfour(int_to_hex(params['IntegralCoeff_2'])) # Word 10
    data += flipfour(params['Message_ID']) # Word 11

    for i in range(len(params['Iset_1'])): # Words 12-111
        data += flipfour(int_to_hex(cnv.conv_I_mA_to_N(params['Iset_1'][i])))
    for i in range(len(params['Iset_2'])):
        data += flipfour(int_to_hex(cnv.conv_I_mA_to_N(params['Iset_2'][i]))) # Words 112-211

    CRC_input = []
    for i in range(1,int(len(data)/4)):
        CRC_input.append(data[4*i:4*i+4])

    CRC = crc(CRC_input)
    data += CRC # Word 212
```

```
    return bytearray.fromhex(data)


# ---- Decoding functions

def decode_STATE(state):
    st = flipfour(state)
    if st == '0000':
        status = "All ok."
    elif st == '0001':
        status = "SD Card reading/writing error (SD_ERR)."
    elif st == '0002':
        status = "Command error (UART_ERR)."
    elif st == '0004':
        status = "Wrong parameter value error (UART_DECODE_ERR)."
    elif st == '0008':
        status = "Laser 1: TEC driver overheat (TEC1_ERR)."
    elif st == '0010':
        status = "Laser 2: TEC driver overheat (TEC2_ERR)."
    elif st == '0020':
        status = "Resetting system error (DEFAULT_ERR)."
    elif st == '0040':
        status = "File deletion error (REMOVE_ERR)."
    else:
        status = "Unknown or reserved error."
    return status


def decode_DATA(dh):
    def get_word(s,num):
        return flipfour(s[num*2*2:num*2*2+4])
    def get_int_word(s,num):
        return int(get_word(s,num),16)

    data = {}
    data['datetime'] = datetime.now()
    data['Header']=get_word(dh,0)
    data['I1'] = []
    for i in range(1,I_POINTS_GET+1):
        data['I1'].append(cnv.conv_I_N_to_mA(get_int_word(dh,i)))
```

```
    data['I2'] = []
    for i in range(I_POINTS_GET+1,I_POINTS_GET*2+1):
        data['I2'].append(cnv.conv_I_N_to_mA(get_int_word(dh,i)))
    data['TO_LSB']=get_int_word(dh,201)
    data['TO_MSB']=get_int_word(dh,202)
    data['Temp_1']=cnv.conv_T_N_to_C(get_int_word(dh,203))
    data['Temp_2']=cnv.conv_T_N_to_C(get_int_word(dh,204))
    data['Temp_Ext_1']=cnv.conv_TExt_N_to_C(get_int_word(dh,205))
    data['Temp_Ext_2']=cnv.conv_TExt_N_to_C(get_int_word(dh,206))
    data['MON_3V3']=cnv.conv_U3V3_N_to_V(get_int_word(dh,207))
    data['MON_5V1']=cnv.conv_U5V_N_to_V(get_int_word(dh,208))
    data['MON_5V2']=cnv.conv_U5V_N_to_V(get_int_word(dh,209))
    data['MON_7V0']=cnv.conv_U7V_N_to_V(get_int_word(dh,210))
    data['Message_ID']=get_word(dh,211) # Last received command
    data['CRC']= get_word(dh,212)

    return data
```

device_conversion.py

```
import math

# ---- Conversion functions

VREF = 2.5 # Volts

R1 = 10000 # Ohm
R2 = 2200 # Ohm
R3 = 27000 # Ohm
R4 = 30000 # Ohm
R5 = 27000 # Ohm
R6 = 56000 # Ohm

RREF = 28.7 # Ohm (current-setting resistor) @1550 nm - 28.7 Ohm; @840 nm - 10 Ohm

R7 = 22000 # Ohm
R8 = 22000 # Ohm
R9 = 5100 # Ohm
R10 = 180000 # Ohm
```

```
def conv_T_C_to_N(T):
    Rt = 10000 * math.exp( 3900/(T+273) - 3900/298 )
    U = VREF/(R5*(R3+R4)) * ( R1*R4*(R5+R6) - Rt*(R3*R6-R4*R5) ) / (Rt+R1)
    N = int(U * 65535 / VREF)
    if N<0 or N>65535:
        print("Error converting T=" + str(T) + " to N=" + str(N) + ". N should be within [0, 65535]. Returning
N=0.")
    return N

def conv_T_N_to_C(N):
    U = N*VREF/65535 # Volts
    Rt = R1 * (VREF*R4*(R5+R6) - U*R5*(R3+R4)) / (U*R5*(R3+R4) + VREF*R3*R6 - VREF*R4*R5) # Ohm
    T = 1 / (1/298 + 1/3900 * math.log(Rt/10000)) - 273 # In Celsius
    return T

def conv_TExt_N_to_C(N):
    U = N*VREF/4095*1/(1+100000/R10) + VREF*R9/(R8+R9) # Volts
    Rt = R7*U/(VREF-U) # Ohm
    T = 1 / (1/298 + 1/3455 * math.log(Rt/10000)) - 273 # In Celsius
    return T

def conv_I_mA_to_N(I):
    N = int(65535/2000 * RREF * I) # I in mA
    if N<0 or N>65535:
        print("Error converting I=" + str(I) + " to N=" + str(N) + ". N should be within [0, 65535]. Returning N=0.")
        N=0
    return N

def conv_I_N_to_mA(N):
    return N*2.5/(65535*4.4) - 1/20.4 # I in mA


def conv_U3V3_N_to_V(u_int):
    return u_int * 1.221 * 0.001 # Volts

def conv_U5V_N_to_V(u_int):
    return u_int * 1.8315 * 0.001 # Volts

def conv_U7V_N_to_V(u_int):
    return u_int * 6.72 * 0.001 # Volts
```

gui.py

```
import PySimpleGUI as sg

#### ---- GUI Constants

WINDOW_TITLE = 'Модуль управления лазерной схемой оптического смесителя (Отдел радиофотоники МФТИ)'
WINDOW_SIZE = [1200, 800]

SET_BUTTON_TEXT = 'Задать'

SET_TEMPERATURE_TEXT_1 = 'Установка температуры лазера 1 (C):'
SET_TEMPERATURE_TEXT_2 = 'Установка температуры лазера 2 (C):'
SET_CURRENT_TEXT_1 = 'Управляющий ток лазера 1 (15-60 мА):'
SET_CURRENT_TEXT_2 = 'Управляющий ток лазера 2 (15-60 мА):'
SET_TEXT_WIDTH = 34
SET_INPUT_WIDTH = 5

GRAPH_POINTS_NUMBER = 100 # Number of most recent data points shown on charts
GRAPH_CANVAS_SIZE = (540,240)
GRAPH_BG_COLOR = '#303030'
GRAPH_SIGN_AXES_COLOR = 'orange'

GRAPH_T_MIN = 0 # Celsius
GRAPH_T_MAX = 50 # Celsius
GRAPH_I_MIN = 0.0 # mA
GRAPH_I_MAX = 1.0 # mA



READ_TEMPERATURE_TEXT = 'Температура лазера'
READ_CURRENT_TEXT = 'Ток фотодиода'

VOLTAGE_TEXT_WIDTH = 15



#### ---- Setting GUI

def setup_gui(params):
```

```
  sg.theme("DarkBlue12")

  layout_input_col1 = [[sg.Text(SET_TEMPERATURE_TEXT_1, size=(SET_TEXT_WIDTH,1)),
                    sg.Input(params['Temp_1'], size=(SET_INPUT_WIDTH,1), key='-InputT1-'),
                    sg.Button(SET_BUTTON_TEXT, key='-SetT1-')],

                   [sg.Text(SET_CURRENT_TEXT_1, size=(SET_TEXT_WIDTH,1)),
                    sg.Input(params['Iset_1'][0], size=(SET_INPUT_WIDTH,1), key='-InputI1-'),
                    sg.Button(SET_BUTTON_TEXT, key='-SetI1-')],

                   [sg.HSeparator(pad=(1,20))],

                   [sg.Push(), sg.Text(READ_TEMPERATURE_TEXT+' 1: ', key='-TOUT_1-')],

                   [sg.Graph(canvas_size=GRAPH_CANVAS_SIZE, graph_bottom_left=(0, GRAPH_T_MIN),
graph_top_right=(GRAPH_POINTS_NUMBER, GRAPH_T_MAX),
                    background_color=GRAPH_BG_COLOR, enable_events=False, drag_submits=False,
key='-GraphT1-')],

                   # [sg.HSeparator(pad=(10,15), color=sg.theme_background_color())],

                   [sg.Push(), sg.Text(READ_CURRENT_TEXT+' 1: ', pad=(5, (20,5)), key='-IOUT_1-')],

                   [sg.Graph(canvas_size=GRAPH_CANVAS_SIZE, graph_bottom_left=(0, GRAPH_I_MIN),
graph_top_right=(GRAPH_POINTS_NUMBER, GRAPH_I_MAX),
                    background_color=GRAPH_BG_COLOR, enable_events=False, drag_submits=False,
key='-GraphI1-')]]


  layout_input_col2 = [[sg.Text(SET_TEMPERATURE_TEXT_2, size=(SET_TEXT_WIDTH,1)),
                    sg.Input(params['Temp_2'], size=(SET_INPUT_WIDTH,1), key='-InputT2-'),
                    sg.Button(SET_BUTTON_TEXT, key='-SetT2-')],

                   [sg.Text(SET_CURRENT_TEXT_2, size=(SET_TEXT_WIDTH,1)),
                    sg.Input(params['Iset_2'][0], size=(SET_INPUT_WIDTH,1), key='-InputI2-'),
                    sg.Button(SET_BUTTON_TEXT, key='-SetI2-')],

                   [sg.HSeparator(pad=(1,20))],

                   [sg.Push(), sg.Text(READ_TEMPERATURE_TEXT+' 2: ', key='-TOUT_2-')],
```

```
            [sg.Graph(canvas_size=GRAPH_CANVAS_SIZE, graph_bottom_left=(0, GRAPH_T_MIN),
graph_top_right=(GRAPH_POINTS_NUMBER, GRAPH_T_MAX),
              background_color=GRAPH_BG_COLOR, enable_events=False, drag_submits=False,
key='-GraphT2-')],

            # [sg.HSeparator(pad=(10,15), color=sg.theme_background_color())],

            [sg.Push(), sg.Text(READ_CURRENT_TEXT+' 2: ', pad=(5, (20,5)), key='-IOUT_2-')],

            [sg.Graph(canvas_size=GRAPH_CANVAS_SIZE, graph_bottom_left=(0, GRAPH_I_MIN),
graph_top_right=(GRAPH_POINTS_NUMBER, GRAPH_I_MAX),
              background_color=GRAPH_BG_COLOR, enable_events=False, drag_submits=False,
key='-GraphI2-')]]


  layout = [[sg.Column(layout_input_col1), sg.VSeparator(), sg.Column(layout_input_col2)],

        [sg.HSeparator(pad=(25,10))],

        [sg.Text('', size=(7,1)),
          sg.Text('T терм 1:', size=(VOLTAGE_TEXT_WIDTH,1), key='-TTerm1-'), sg.Text('T терм 2:',
size=(VOLTAGE_TEXT_WIDTH,1), key='-TTerm2-'),
          sg.Text('3V3:', size=(VOLTAGE_TEXT_WIDTH,1), key='-3V3-'), sg.Text('5V1:',
size=(VOLTAGE_TEXT_WIDTH,1), key='-5V1-'),
          sg.Text('5V2:', size=(VOLTAGE_TEXT_WIDTH,1), key='-5V2-'), sg.Text('7V0:',
size=(VOLTAGE_TEXT_WIDTH,1), key='-7V0-'),
          sg.Push(), sg.Text('', key='-DateTime-', pad=(1,10)),
          sg.Text('', size=(10,1))],

        [sg.Exit('Выход', pad=(1,5), size=(10,1), key='-EXIT-')]]


  return sg.Window(WINDOW_TITLE,
            layout,  finalize=True, element_justification='c', size=WINDOW_SIZE)


def sign_axes(window):
  signs_dict = {}
  signs_dict['-GraphT1-'] = \
```

```
        (window['-GraphT1-'].draw_text(text=str(GRAPH_T_MIN)+' C', location=(3,
GRAPH_T_MIN+(GRAPH_T_MAX-GRAPH_T_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR),
         window['-GraphT1-'].draw_text(text=str(GRAPH_T_MAX)+' C', location=(3,
GRAPH_T_MAX-(GRAPH_T_MAX-GRAPH_T_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR))
    signs_dict['-GraphI1-'] = \
        (window['-GraphI1-'].draw_text(text=str(GRAPH_I_MIN)+' мA', location=(4,
GRAPH_I_MIN+(GRAPH_I_MAX-GRAPH_I_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR),
         window['-GraphI1-'].draw_text(text=str(GRAPH_I_MAX)+' мA', location=(4,
GRAPH_I_MAX-(GRAPH_I_MAX-GRAPH_I_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR))
    signs_dict['-GraphT2-'] = \
        (window['-GraphT2-'].draw_text(text=str(GRAPH_T_MIN)+' C', location=(3,
GRAPH_T_MIN+(GRAPH_T_MAX-GRAPH_T_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR),
         window['-GraphT2-'].draw_text(text=str(GRAPH_T_MAX)+' C', location=(3,
GRAPH_T_MAX-(GRAPH_T_MAX-GRAPH_T_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR))
    signs_dict['-GraphI2-'] = \
        (window['-GraphI2-'].draw_text(text=str(GRAPH_I_MIN)+' мA', location=(4,
GRAPH_I_MIN+(GRAPH_I_MAX-GRAPH_I_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR),
         window['-GraphI2-'].draw_text(text=str(GRAPH_I_MAX)+' мA', location=(4,
GRAPH_I_MAX-(GRAPH_I_MAX-GRAPH_I_MIN)*0.05), color=GRAPH_SIGN_AXES_COLOR))
    return signs_dict
```