Титульник

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью проекта было создание бюджетного портативного медицинского пирометра с возможностью автоматической отправки данных на удаленный сервер для последующего анализа и хранения. Актуальность выбранной темы объясняется ростом числа устройств интернета вещей в области здравоохранения. Внедрение IoT технологий позволяет повысить продуктивность персонала, ускорить процессы сбора, передачи и анализа данных, снизить влияние человеческого фактора и уменьшить стоимость лечения.

Поставленные задачи:

- Выбор оптимального пирометрического датчика
- Разработка технической части устройства
- Разработка алгоритма и написание программы
- Реализация отправки данных в облачный сервис для хранения и обработки информации
- Сборка устройства, программирование, тестирование

Анализ рынка показал, что устройств подобного типа на рынке практически нет. Существует небольшое число устройств с возможностью передачи данных по Bluetooth на мобильной устройство, находящееся рядом, для хранения и анализа этих данных в приложении на устройстве. Единственное устройство с передачей данных по WiFi – устройство компании ПрофКиП – СтражниК. Однако данный прибор отправляет данные только в приложение на мобильном устройстве для мониторинга. В таблице ниже можно увидеть сравнения некоторых приборов с разрабатываемым.

Таблица 1. Сравнение устройства с аналогами

	ПрофКиП СтражниК	**HTi HT-806**	**AndesFit ADF-B38A**	**Viatom by Lepu AOJ-20A**	**Разрабатываемое устройство**
Диапазон, °C	+ 30 …+ 42	- 20 …+ 550	+ 35 …+ 42	+ 32 …+ 42	- 40 …+ 85
Точность, °C	± 0.3	± 2	± 0.3	± 0.2	± 0.1
Питание	От сети	Автономное	Автономное	Автономное	Автономное
Интерфейс	WiFi	Bluetooth LE	Bluetooth LE	Bluetooth LE	WiFi
Тип	Медицинский	Промышленный	Медицинский	Медицинский	Медицинский
Цена, руб	35 000	2 850	4748	2 067	2 356

Проект нацелен на компании с большим числом сотрудников, общественные мероприятия, медицинские учреждения, а также подходит для частного использования. Использование устройства в общественных местах, в пропускных пунктах, позволит не только своевременно выявлять потенциально зараженных людей, но и собирать статистику по району, городу в целом, для анализа и выявления возможных очагов заражений в будущем.

ОПИСАНИЕ РЕАЛИЗАЦИИ

В данном разделе будут подробно рассмотрены выбранные платы и устройства, используемые в проекте, а также причины их выбора. Правильный подбор составляющих частей устройства необходим для его корректной работы. Важно обратить внимание на совместимость плат, а также на необходимое напряжение питания каждой платы. Неправильно поданное питание может привести к некорректной работе устройства или к выходу его из строя.

Плата на базе микроконтроллера ESP32

Для обработки данных с пирометрического датчика, а также для обеспечения коммуникации между отдельными частями устройства, будет использоваться микроконтроллер Espressif ESP32.

Для проекта была необходима доступная универсальная плата с встроенным модулем Wi-Fi. Выбор был сделан в пользу ESP32 WEMOS D1 R32. Эта плата выполнена в формате Arduino Uno, что обеспечивает ее совместимость с большим количеством модулей, а в ее основе лежит WiFi Bluetooth модуль WROOM-32, который содержит микроконтроллер ESP32. Ниже представлены изображения платы, а также основные характеристики из официальной документации к устройству [1] и с сайта продавца [2].

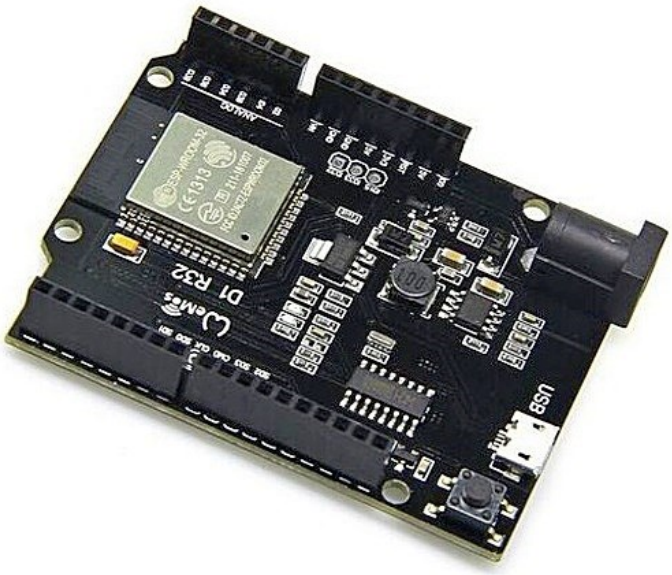

Рисунок 1 - Внешний вид платы WEMOS D1 R32

Таблица 2. Основные характеристики WEMOS D1 R32

Микроконтроллер	Tensilica LX6 Espressif ESP32
Рабочее напряжение Рабочий ток Напряжение питания	3.3 В 80 мА 5-12 В
Wi-Fi Bluetooth	802.11 b/g/n (802.11n up to 150 Mbps) Bluetooth v4.2 BR/EDR, Bluetooth LE
Flash-память	4 Мб
Поддерживаемые интерфейсы	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface (TWAI®), compatible with ISO11898-1 (CAN Specification 2.0)
Рабочие температуры	-40 ºC … + 85 ºC

Питание платы осуществляется через разъем micro-USB или через штекер питания напряжением 5-12 В. Подключение к персональному компьютеру осуществляется также через разъем micro-USB. Рабочее напряжение контактов 3.3 В, поэтому плата может не работать с некоторыми платами расширения Arduino. Всего на плате имеется 41 разъем, которые используются для подключения периферийных устройств. На рисунке (Рисунок 2) изображено расположение контактов на плате и возможные режимы работы каждого из них. Питание периферийных устройств может осуществляться напряжениями 5В и 3.3В.

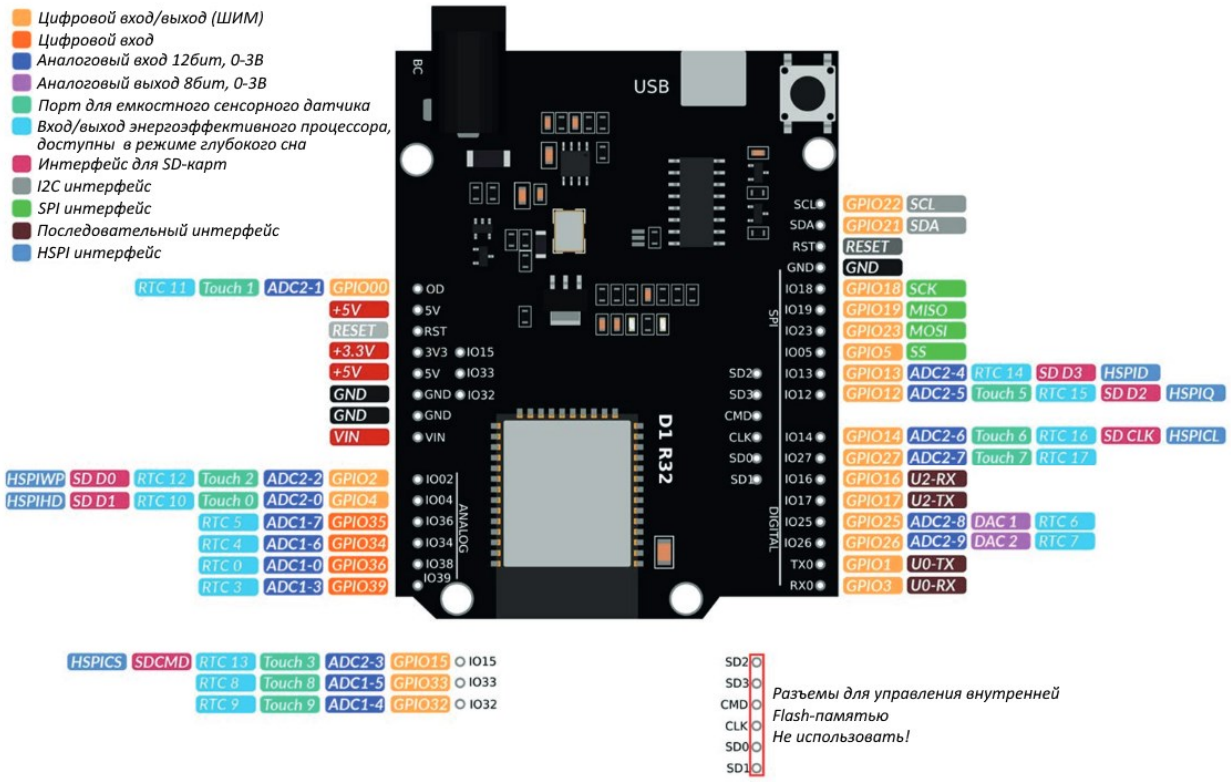

Рисунок 2 – Назначение выводов платы

Выбор пирометрического датчика

Температурный датчик выбирался таким образом, чтобы удовлетворить ряду условий: низкая цена, высокая точность в диапазоне измеряемых температур, доступность на рынке. Под эти критерии подходит датчик MLX90614 компании Melexis. Внешний вид датчика показан на рисунке ниже.

Рисунок 3 – Изображение датчика MLX90614

Информация, приведенная ниже, взята из официальной документации к датчику [3] и с официального сайта производителя [4]. Датчик выполняется в различных конфигурациях (Рисунок 4). Для медицинских применений целесообразно выбрать датчик, откалиброванный с высокой точностью специально для этих целей. Остальными, менее важными параметрами, можно пожертвовать, для снижения стоимости конечного продукта. Выберем датчик MLX90614ESF-DAA-000-TU.

Part No.	Temperature Code	Package Code	- Option Code	Standard part	Packing form
MLX90614	E (-40°C...85°C) K (-40°C...125°C)	SF (TO-39)	- X X X (1) (2) (3)	-000	-TU

(1) Supply Voltage/ Accuracy	(2) Number of thermopiles:	(3) Package options:
A - 5V	A – single zone	A – Standard package
B - 3V	B – dual zone	B – Reserved
C - Reserved	C – gradient compensated*	C – 35° FOV
D - 3V medical accuracy		D/E – Reserved
		F – 10° FOV
		G – Reserved
		H – 12° FOV (refractive lens)
		I – 5° FOV
		K – 13°FOV

Рисунок 4 – Варианты исполнения датчика

Датчики MLX90614 предназначены для бесконтактного измерения температуры. Внутри стандартного корпуса TO-39 расположены ИК-детектор MLX81101 и блок обработки сигналов MLX90302, разработанный специально для обработки сигналов с ИК-детекторов. MLX90302 включает в себя усилитель, 17-ти битный АЦП, блок цифровой обработки сигналов, блок ШИМ модуляции и устройство управления (Рисунок 5).

Датчик содержит четыре контакта (Рисунок 6). Два из них предназначены для подачи питания: VDD (питание) и VSS (земля). Другие два используются для передачи информации по шине I2C. При работе в режиме несовместимым с I2C контакт SDA/PWM передает измеренную температуру в 10-ти битном ШИМ-модулированном сигнале.

Значение коэффициента излучения в датчике было выбрано на основании исследований в книге [5]. Коэффициент излучения кожи выбран 0.97.

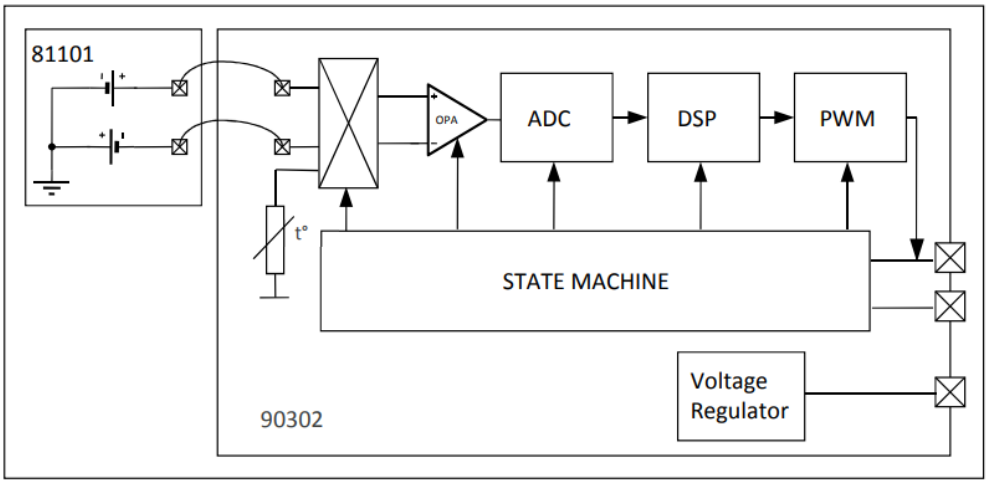

Рисунок 5 – Блок-диаграмма MLX90614

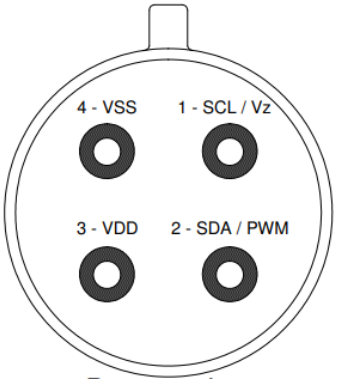

Рисунок 6 – Расположение контактов датчика

Основные характеристики датчика отображены в таблице ниже.

Таблица 3. Основные характеристики MLX90614

Рабочее напряжение	3В
Рабочее напряжение (предельное)	3.6В
Рабочий ток	1.3 мА
Диапазон измеряемых температур (объект)	-40 ºС … + 85 ºС
Диапазон измеряемых температур (датчик)	-70 ºС … + 380 ºС
Точность измерений (объект)	± 0.1 ºС
Точность измерений (датчик)	± 0.5 ºС
Рабочая температура	-40 ºС … + 85 ºС
Температура хранения	-40 ºС … + 125 ºС
АЦП	17 бит
ШИМ	10 бит

LCD дисплей

Для отображения информации о работе устройства и измеряемой температуры будем использовать LCD дисплей LCD1602A. Он способен отображать 2 строки по 16 символов каждая. Каждый символ может иметь размеры 5 на 8 точек. Питание осуществляется напряжение 5В. Основные характеристики дисплея отображены в таблице ниже. Внешний вид дисплея представлен на рисунке (Рисунок 7). Основные характеристики устройства можно видеть в таблице ниже. Информация приведена из документации к устройству [6].

Таблица 4. Основные характеристики LCD1602A

Рабочее напряжение Рабочее напряжение (предельное) Рабочий ток	5В 5.5В 1.1 мА
Рабочая температура Температура хранения	0 ºС … + 50 ºС -10 ºС … + 60 ºС
Тип подсветки Цвет подсветки	Боковая Желтая

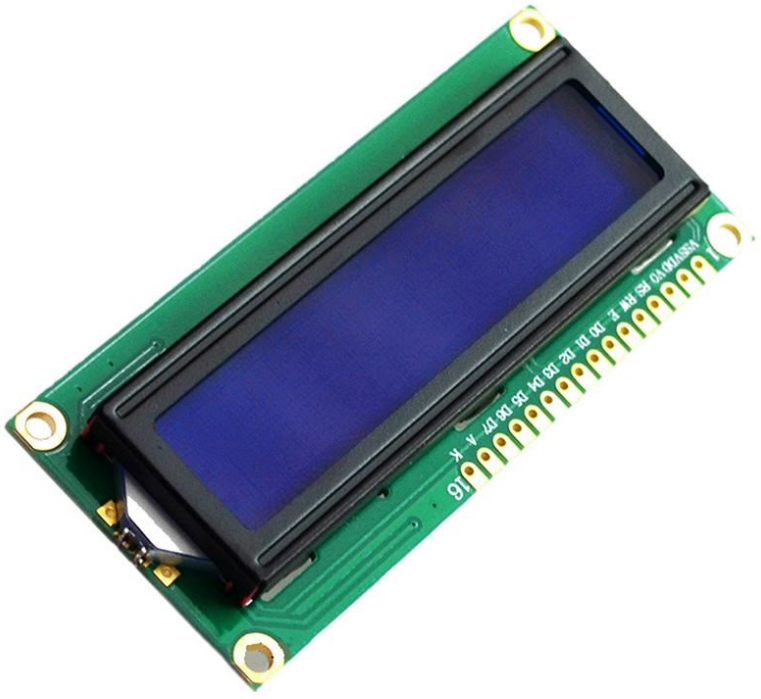

Рисунок 7 – Внешний вид LCD1602A

Для удобства использования дисплея, воспользуемся конвертером IIC в I2C и подключим дисплей к шине I2C вместе с датчиком MLX90614. Таким образом, для подключения дисплея нужно всего четыре провода, вместо шестнадцати, два из которых отвечают за питание и два за передачу информации по шине I2C. Внешний вид конвертера и основные характеристики отображены ниже.

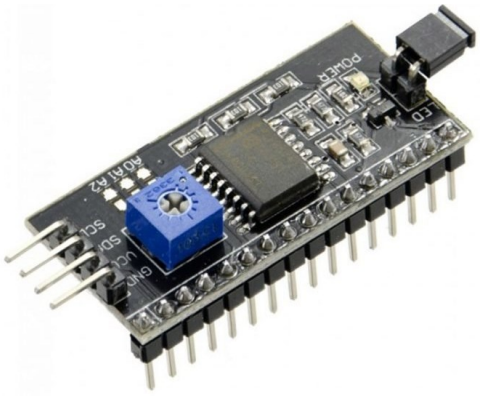

Рисунок 8 – Внешний вид конвертера IIC/I2C

Таблица 5. Основные характеристики конвертера IIC/I2C

Рабочее напряжение	5В
Регулировка контраста Регулировка подсветки	Есть, потенциометр Есть, программная или джампер
Поддержка интерфейсов	IIC, I2C, TWI

Периферийные устройства

Для начала измерения требуется получить сигнал от оператора или от срабатывания датчика или таймера. В данной работе, для демонстрации, используется кнопка, после нажатия которой прибор производит измерение температуры. По мимо кнопки, может использоваться датчик приближения, движения, или другие устройства.

Для оповещения об успешном или ошибочном измерении, об успешности отправки данных на сервер, и о повышении температуры тела, используется зуммер. Частота и количество сигналов зависят от результата измерения или отправки данных.

Сборка устройства

Как уже говорилось ранее, дисплей и ИК-датчик подключаются к одной шине I2C. Обращение к ним происходит по индивидуальным адресам. Для MLX90614 это 5A HEX, а для LCD1602A – 27 HEX.

Кнопка, как и зуммер подключены к стандартным GPIO разъемам. Кнопка подтянута на землю, стандартный уровень – низкий. При нажатии, на входе GPIO появляется высокий уровень, который регистрируется микроконтроллером. Зуммер управляется программой через цифровой выход микроконтроллера. На рисунке (Рисунок 9) изображена схема подключения устройств к микроконтроллеру, а на рисунке (Рисунок 10) изображен внешний вид прототипа устройства.

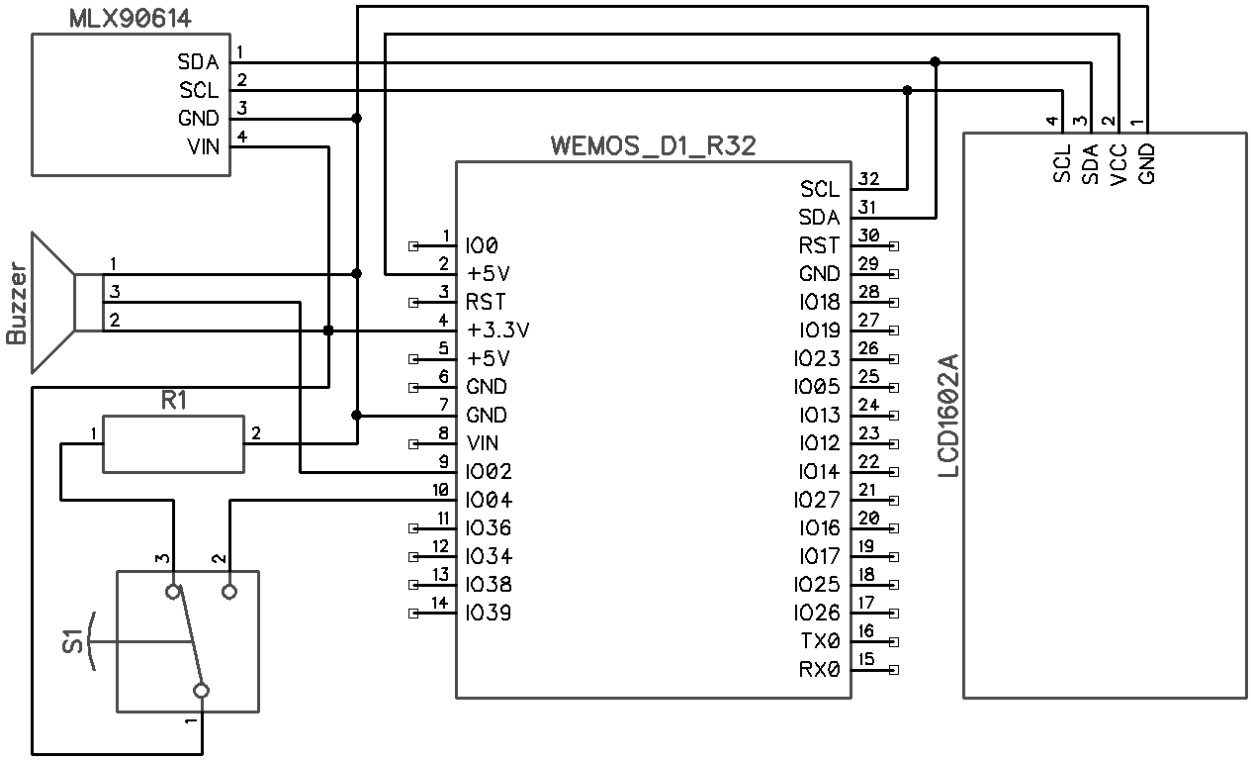

Рисунок 9 – Схема подключения устройств

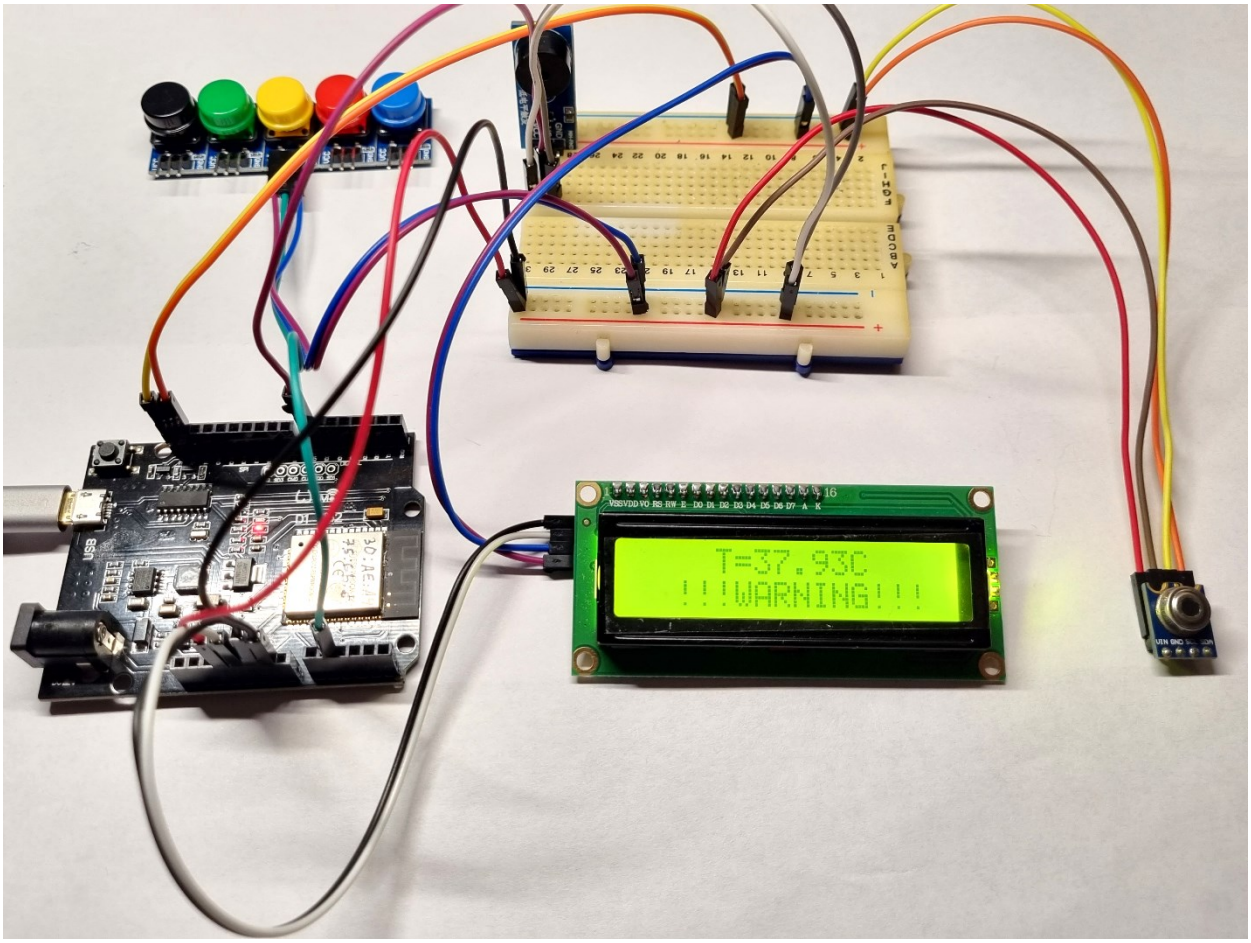

Рисунок 10 – Внешний вид прототипа устройства

РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ

Алгоритм работы устройства

Укрупненная блок-схема алгоритма работы программы представлена на рисунке (Рисунок 11). После включение устройства происходит инициализация микроконтроллера и всех подключенных устройств. Далее начинается работа основного цикла. Каждый раз, в начале цикла, проверяется наличие подключения к сети Wi-Fi и циклично выполняются попытки подключения к выбранной сети в случае отсутствия подключения в текущий момент времени. Пока устройство не подключится к сети Wi-Fi, программа не продолжит свою работу. Данные о сети, необходимые для подключения, находятся в отдельном файле, для безопасности (ПРИЛОЖЕНИЕ Б). Информация о процессе инициализации микроконтроллера и модулей, а также о подключении к сети Wi-Fi, отображается на дисплее и выводится в последовательный порт.

При успешном подключении, программа будет ждать сигнала к началу измерения температуры. В зависимости, от предпочтений заказчика, сигнал к началу измерения может подаваться любым способом. После получения сигнала, устройство измеряет температуру биообъекта, выводит ее на дисплей и отправляет данные на удаленный сервер, для дальнейшей обработки и хранения данных. В случае повышенной температуры тела, предусмотрена звуковая индикация и предупреждение на дисплее. На дисплее, помимо текущей температуры, отображается информация об успешной или ошибочной отправке данных на сервер. Полный текст программы находится в приложении (ПРИЛОЖЕНИЕ А).

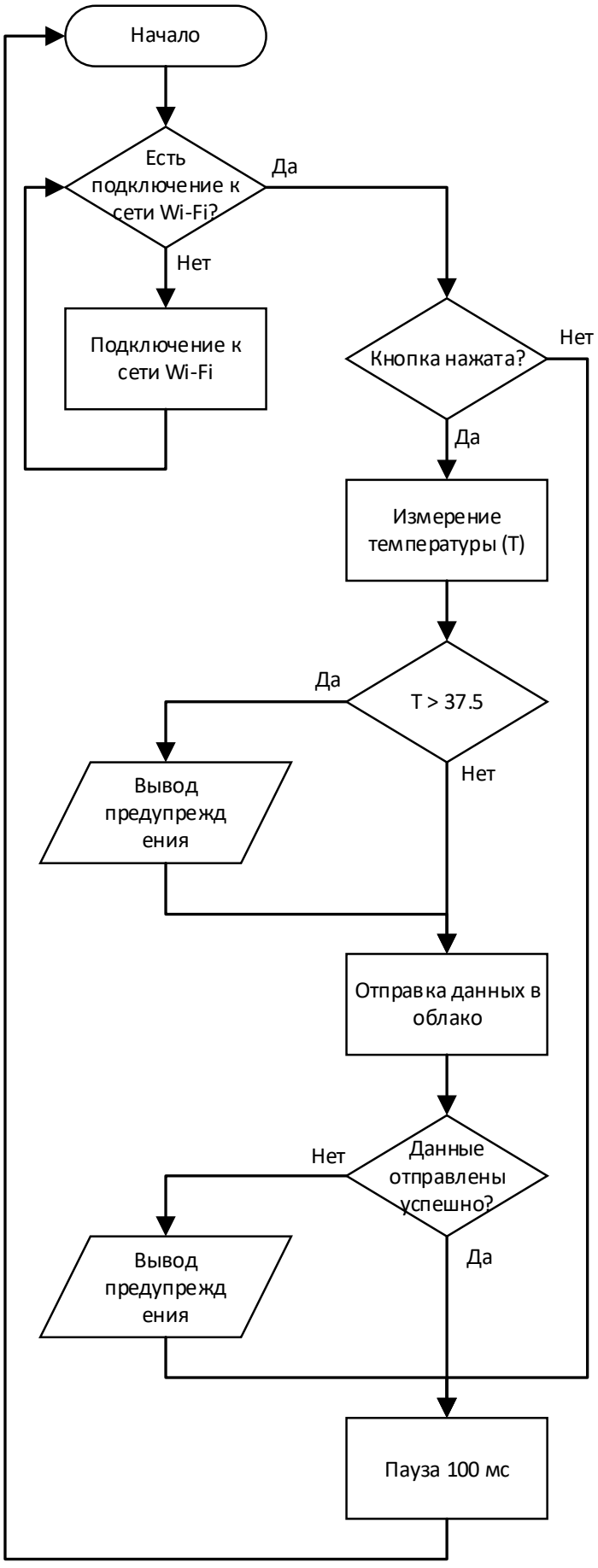

Рисунок 11 – Блок-схема алгоритма работы программы

Отправка данных на сервер

Отправка данных на удаленный сервер осуществляется средствами протокола MQTT. Данный протокол был выбран из-за своих достоинств:

- простота реализации;
- обеспечивание формы качества обслуживания;
- эффективность с точки зрения пропускной способности;
- платформенно-независимая;
- постоянное отслеживание сеанса;
- решение проблем безопасности.

Структура протокола следующая: клиента, который передает сообщение, называют издателем, а клиента, получающего сообщение – подписчиком, между ними находится брокер MQTT, который несет ответственность за соединение клиентов и фильтрацию поступающих данных. Клиенты публикуют и/или подписываются на темы, управляемые брокером MQTT.

Помимо описанных преимуществ протокола, также можно выделить его временную независимость, т.е. сообщение, опубликованное одним клиентом, может быть прочитано в любое время. Благодаря тому, что брокер является руководящим органом между издателями и потребителями, нет необходимости напрямую идентифицировать издателя и потребителя на основе физических данных, что очень полезно для IoT систем, поскольку физический идентификатор может быть неизвестным или общим [7].

В данной работе, в качестве демонстрации отправка данных будет осуществляться на сервера сервиса ThingSpeak.

ThingSpeak

Сервис ThingSpeak (https://thingspeak.com) – открытая аналитическая платформа для проектов IoT, включающая в себя сбор данных с датчиков в реальном времени, обработку этих данных, их визуализацию и использование в приложениях и плагинах.

ThingSpeak позволяет обрабатывать до 8 различных типов данных в одном канале. Каждый тип данных занимает свое поле. Экран настройки канала показан на рисунке (Рисунок 12). Для наших целей достаточно два поля: для измеряемой температуры и для счетчика потенциально заболевших людей. При необходимости, можно увеличить количество полей, добавив другие переменные. Помимо полей, содержащих значения, сервис предлагает возможность добавить описание для канала, его территориальное расположение, и ссылки на различные ресурсы.

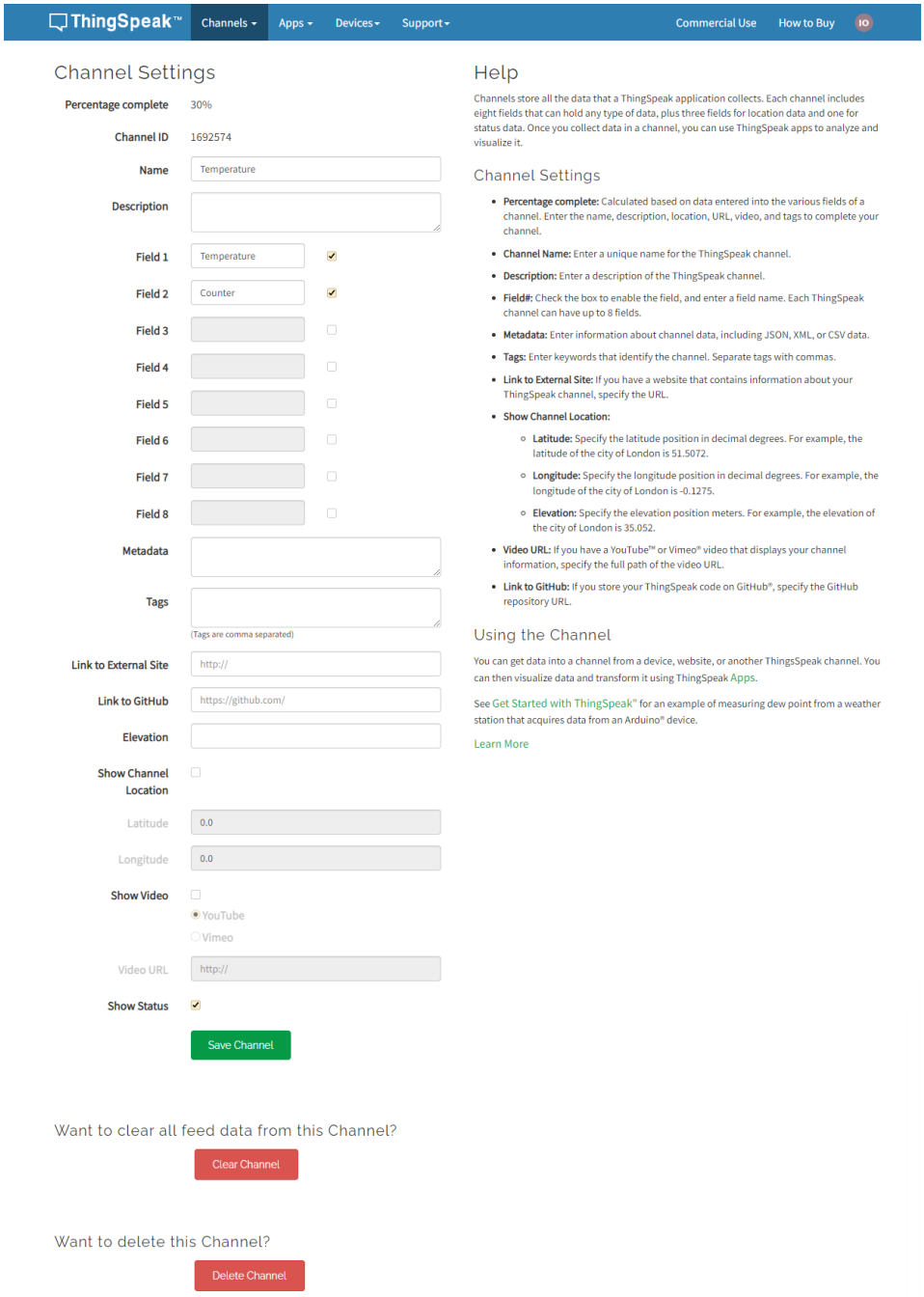

Рисунок 12 – Настройка канала ThingSpeak

Каждый канал имеет уникальный идентификационный номер (Channel ID), а также ключи для разрешения записи и чтения информации в канале (Write API Key и Read API Keys). Канал может быть как приватным, так и открытым, давая возможность другим пользователям следить за изменениями в канале. Записывать данные в канале или читать данные из него могут все пользователи, у которых имеется номер канала и соответствующие ключи.

Платформа работает с большим количеством компаний, производящих аппаратное и программное обеспечение для применений в IoT. В частности, сервис имеет совместимость с платами на базе ESP32 и со средой программирования плат Arduino IDE, что упрощает отправку данных с устройства на сервер. В проекте используется библиотека ThingSpeak для Arduino, ESP8266 и ESP32. Библиотека упрощает доступ к серверам платформы и позволяет обмениваться данными между устройствами и сервером.

В зависимости от реализуемой задачи, можно настроить отображение принимаемой и обработанной информации в наглядном виде. На рисунке (Рисунок 13) представлен один из множества вариантов отображения информации на платформе. В окне слева отображаются новые принимаемые значения в зависимости от времени измерения. В окне справа отображается число потенциально заболевших людей, прошедших контроль температуры. Эти и другие вычисления возможны благодаря интеграции ThingSpeak и MATLAB. Сервис предоставляет возможность использовать пакет MATLAB для анализа данных. Расчет потенциально заболевших людей происходит в пакете MATLAB, после чего результирующие данные могут быть выведены в отдельное поле канала. Полный текст программы, используемой для определения числа заболевших приведен в приложении (ПРИЛОЖЕНИЕ В).

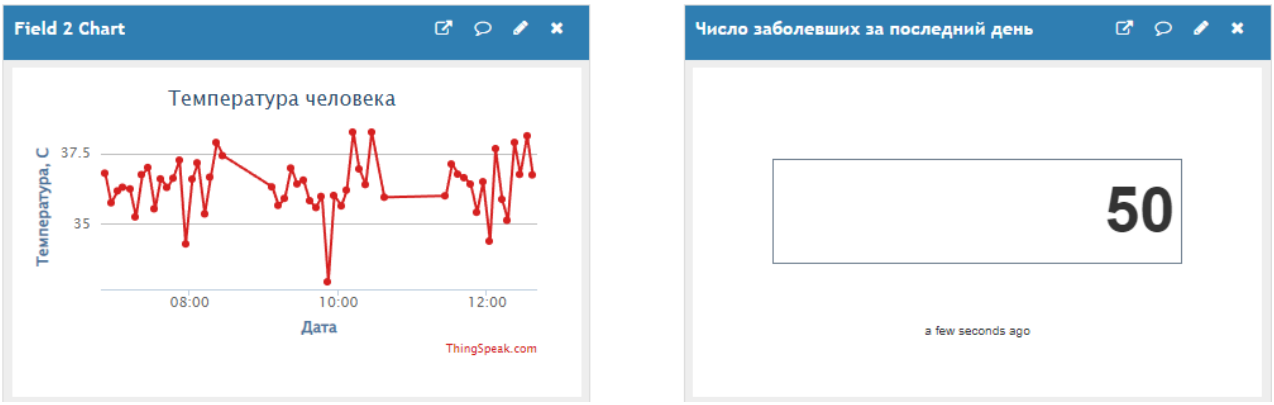

Рисунок 13 – Представление данных в ThingSpeak

К сожалению, сервис ThingSpeak не позволяет отправлять данные чаще, чем раз в 15 секунд без платной подписки. Он был выбран в качестве демонстрации возможностей хранения и обработки данных на удаленном сервере. Выбор сервера для отправки данных остается за заказчиком.

РЕЗУЛЬТАТЫ РАБОТЫ

Выполненный прототип незначительно отличается от описанного выше. Это связано с плохой доступностью и высокой ценой электронных компонентов на момент выполнения работы (весна 2022 года). Основное отличие – модель датчика MLX90614. Как было сказано ранее, рекомендуется использоваться датчик MLX90614ESF-DAA-000-TU, однако ввиду отсутствия его в продаже, для демонстрации работы устройства был выбран MLX90614ESF-BAA-000-TU. Главное отличие датчиков заключается в точности измерения. Последний обладает более низкой точностью (± 0.5 ºC) в необходимом диапазоне измеряемых температур.

Прототип был собран с использованием макетной платы. Это было сделано для удобства тестирования и возможности оперативно менять конфигурацию устройства. Такая сборка не удобна для повседневного использования и менее надежна, поэтому рекомендуется более надежные соединения компонентов в итоговом продукте.

Рассчитаем максимальное потребление тока устройством как сумму потреблений отдельных компонентов. Важно заметить, что большую часть времени устройство будет потреблять значительно меньший ток, так как одновременно активно работать все компоненты будут в редких случаях. Сведем результаты в таблицу.

Таблица 6. Расчет потребления тока

Компонент	Максимальное потребление, мА
WEMOS D1 R32	250
LCD1602A	200
MLX90614	25
Зуммер	20
Итог	495

Рассчитаем стоимость прототипа устройства и сведем данные в таблицу.

Таблица 7. Смета компонентов

Компонент	Цена, руб
WEMOS D1 R32	805
LCD1602A	340
Зуммер	175
MLX90614	910
Кнопки	126
Итог	2356

Из таблицы можно заметить, что основные элементы устройства – плата микроконтроллера и ИК-датчик – являются самыми дорогими. Общая цена устройства, даже с учетом повышенных цен, получилась не высокой. При этом, использовались оригинальные компоненты, которые обладают высоким качеством и хорошей надежностью.

ЗАКЛЮЧЕНИЕ

В результате выпускной квалификационной работы был разработан прототип портативного медицинского пирометра с функцией автоматической отправки данных на удаленный сервер.

В качестве сервера для хранения и анализа данных была выбрана платформа ThingSpeak. Как уже было отмечено, главным ее недостатком для использования в наших целях является невозможность отправлять данные чаще, чем один раз в 15 секунд. Эта проблема решаема либо покупкой платной подписки на сервис, либо смена платформы ThingSpeak на другую аналогичную, где отсутствуют подобные ограничения. Сервис позволяет хранить большой объем данных и анализировать их средствами интегрированного пакета MatLab. Представлять данные можно в числовой форме, различными индикаторами или в виде графиков. Как уже было сказано, возможно большое количество различных вариантов обработки получаемых данных, в зависимости от поставленных целей. Также, существует возможность добавить несколько пирометров в один канал, передавая данные каждого из них в индивидуальное или общее поле, или добавить другие датчики в сеть и собирать комплексную информацию не только о температуре людей, но и о условиях, в которых проводились эти измерения. Полученные данные потенциально могут помочь в выявлении местных очагов заболеваемости среди населения, выявления и предупреждения распространения новых инфекций. Одно из основных преимуществ хранения и анализа данных на удаленном централизованном сервере – это возможность следить за изменениями в получаемых данных удаленно, с любого устройства, из любой точки планеты с доступом к сети Интернет. Помимо этого, доступ к получаемым данным можно легко предоставить сторонним компаниям, для более качественного анализа большого объема данных с различных предприятий.

В будущем, проект может быть доработан. Основываясь на знаниях и опыте, полученных при разработке прототипа, планируется создание полноценного устройства на отдельной плате, корпуса, сборка и тестирование.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **esp32-wroom-32_datasheet_en.pdf** [Электронный ресурс]. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (дата обращения 08.04.2022)
2. **ESP32 Wemos D1 R32** [Электронный ресурс]. URL: https://arduino-kit.ru/product/esp32-wemos-d1-r32 (дата обращения 11.05.2022)
3. **Datasheet for MLX90614** [Электронный ресурс]. URL: https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90614 (дата обращения 30.03.2022)
4. **Digital plug & play infrared thermometer in a TO-can** [Электронный ресурс]. URL: https://www.melexis.com/en/product/MLX90614/Digital-Plug-Play-Infrared-Thermometer-TO-Can (дата обращения 30.03.2022)
5. **Тепловизионная биомедицинская диагностика** [Книга] / авт. Скрипаль А. В., Сагайдачный А. А. и Усанов Д. А.. - Саратов : Издательство Саратовского университета, 2009. - стр. 16, 96.
6. **I2C_1602_LCD.pdf** [Электронный ресурс]. URL: http://www.handson-tec.com/dataspecs/module/I2C_1602_LCD.pdf (дата обращения 01.06.2022)
7. **Архитектура интернета вещей** [Книга] / авт. Ли П / перев. Райтман М. А.. - Москва : ДМК Пресс, 2019. - стр. 282-297.
8. **Adafruit MLX90614 Library** [Электронный ресурс]. URL: https://www.arduino.cc/reference/en/libraries/adafruit-mlx90614-library/ (дата обращения 08.04.2022)
9. **WiFi** [Электронный ресурс]. URL: https://www.arduino.cc/reference/en/libraries/wifi/ (дата обращения 08.04.2022)
10. **Wire** [Электронный ресурс]. URL: https://www.arduino.cc/reference/en/language/functions/communication/wire/ (дата обращения 08.04.2022)
11. **LiquidCrystal I2C** [Электронный ресурс]. URL: https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/ (дата обращения 08.04.2022)

12. **ThingSpeak** [Электронный ресурс]. URL: https://www.arduino.cc/reference/en/libraries/thingspeak/ (дата обращения 08.04.2022)

ПРИЛОЖЕНИЕ А

```
// Блок 1. Подключение библиотек
#include <Adafruit_MLX90614.h>    // Библиотека для датчика MLX90614 [8]
#include <WiFi.h>                 // Библиотека Wi-Fi [9]
#include "secrets.h"              // Файл с конфиденциальными данными
#include <Wire.h>                 // Библиотека для I2C интерфейса [10]
#include <LiquidCrystal_I2C.h>    // Библиотека для LCD дисплея, подключаемого через I2C интерфейс [1111]
#include "ThingSpeak.h"           // Библиотека для отправки данных в ThingSpeak [12]

// Блок 2. Подключение устройств и систем
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
LiquidCrystal_I2C LCD(0x27, 16, 2); // Определение дисплея
WiFiClient  client;
const int Button = 4;
const int Buzzer = 2;

// Блок 3. Определение констант и переменных
char ssid[] = SECRET_SSID;   // имя сети Wi-Fi
char pass[] = SECRET_PASS; // пароль сети Wi-Fi
unsigned long myChannelNumber = SECRET_CH_ID; // номер канала ThingSpeak
const char * myWriteAPIKey = SECRET_WRITE_APIKEY; // ключ API для записи данных в ThingSpeak
float T_obj = 0; // переменная для хранения измеренной температуры

// Блок 4. Описание функций
void DisplayInit() { // Функция инициализации дисплея
   Serial.print("Display... ");
   LCD.init(); // Инициализация дисплея
```

```
  Wire.beginTransmission(byte(39)); // Начало передачи данных дисплею
  if (Wire.endTransmission() != 0) { // Проверка наличия связи с дисплеем
    Serial.println("ERROR");
    while (1); // Бесконечный цикл в случае ошибки
  }
  Serial.println("OK!");
  LCD.backlight(); // Включение подсветки дисплея
  LCD.display(); // Включение дисплея
  LCD.clear(); // Очистка дисплея
}

void mlxInit() { // Функция инициализации ИК-датчика
  Serial.print("MLX90614... ");
  mlx.begin(); // Инициализация датчика
  if (!mlx.begin()) { // Проверка наличия связи с датчиком
    Serial.println("ERROR");
    LCD.clear();
    LCD.println("MLX90614 ERROR"); // Вывод на дисплей информации об ошибке
    while (1);
  }
  Serial.println("OK!");
}

void WiFiConnect() { // Функция подключения к сети Wi-Fi
  if (WiFi.status() != WL_CONNECTED) { // Проверка статуса подключения
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SECRET_SSID);
    LCD.clear();
    LCD.print("Connecting to");
```

```
        LCD.setCursor(0, 1);
        LCD.print(SECRET_SSID);
        while (WiFi.status() != WL_CONNECTED) {
           WiFi.begin(ssid, pass); // Подключение к сети Wi-Fi
           Serial.print(".");
           delay(5000);
        }
        Serial.println("\nConnected.");
        LCD.clear();
        LCD.setCursor(3, 0);
        LCD.print("Connected");
        }
}

void Buzz(int K){ // Функция воспроизведения звука
  for (byte i = 0; i < K; i++){
    analogWrite(Buzzer, 250);
    delay(300);
    analogWrite(Buzzer, 0);
    delay(100);
  }
}

// Блок 5. Инициализация устройства
void setup() {
    Serial.begin(9600);
    Serial.println("Initialization..");
    DisplayInit();
    LCD.print("Initialization..");
    mlxInit();
```

```
  WiFi.mode(WIFI_STA); // Установка режима работы WI-FI модуля
  pinMode(Button, OUTPUT); // Установка режима подключения кнопки
  ThingSpeak.begin(client); // Инициализация библиотеки ThingSpeak
  WiFiConnect();
  Serial.println("READY!");
}

// Блок 6. Основной цикл
void loop() {

  WiFiConnect();

  if (digitalRead(Button)) { // Проверка нажатия кнопки
    T_obj = mlx.readObjectTempC(); // Измерение температуры и запись в переменную
    while (T_obj > 85 || isnan(T_obj)) { // Проверка правильности измерения
      T_obj = mlx.readObjectTempC();
    }
    LCD.clear();
    LCD.setCursor(4, 0);
    LCD.print("T=");
    LCD.print(T_obj);
    LCD.print("C");
    LCD.setCursor(2, 1);
    if (T_obj > 37.5) { // Проверка повышенной температуры
      LCD.setCursor(2, 1);
      LCD.print("!!!WARNING!!!");
      Buzz(3);
    }
    else Buzz(1);
```

```
      LCD.setCursor(2, 1);
      ThingSpeak.setField(2, T_obj); // Установка данных для отправки в облако
      // Отправка данных в облако и запись в переменную кода результата от-
правки
      int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
      if (x == 200) {
        Serial.println("Channel update successful.");
        LCD.print("Successfull");
      }
      else {
        // Вывод кода ошибки
        Serial.println("Problem updating channel. HTTP error code " + String(x));
        LCD.print("Error: ");
        LCD.print(String(x));
      }
    }
    delay(100); // Задержка 0.1с
}
```

ПРИЛОЖЕНИЕ Б

```
#define SECRET_SSID "XXXXXXXX"     // имя сети Wi-Fi
#define SECRET_PASS "XXXXXXXX"     // пароль сети Wi-Fi

#define SECRET_CH_ID XXXXXXXX                  // номер канала ThingSpeak
#define  SECRET_WRITE_APIKEY  "XXXXXXXX"      // ключ API для записи данных в ThingSpeak
```

ПРИЛОЖЕНИЕ В

```
ChannelID = XXXXXXX;        // номер канала
TemperatureID = 2;          // номер поля, содержащего измерения
CounterID = 3;              // номер поля, содержащий счетчик
readAPIKey = ‘XXXXXX’;      // код для чтения из канала
WriteAPIKey = ‘XXXXXX’;     // код для записи в канал
C = 0;                      // счетная переменная
// чтение данных за последний день из поля с измерениями
Temperature = thingSpeakRead(ChannelID,'Fields',TemperatureID,'NumMinutes',
1440,'ReadKey',readAPIKey);
// подсчет значений, выходящих за установленный предел
for i = 1:length(Temperature)
        if Temperature(i) >= 37.5
                C = C + 1;
        end
end
// запись данных в поле с счетчиком
thingSpeakWrite(ChannelID,C,'Fields',CounterID,'WriteKey',WriteAPIKey);
```