Министерство науки и образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа компьютерных наук и электроники
Кафедра «Информационно-измерительная техника»

Разработка датчика бесконтактного измерения температуры с передачей параметров по беспроводному интерфейсу

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ
по дисциплине: «Программное обеспечение измерительных процессов»
ЮУрГУ-12.03.01.2020.303 ПЗ КР
ЮУрГУ-12.03.01.2020.296 ПЗ КР
ЮУрГУ-12.03.01.2020.310 ПЗ КР

Нормоконтролер (доцент)
_____/ С.В. Колодий
« » _____ 2020 г.

Руководитель (доцент)
_____/ С.В. Колодий
« » _____ 2020 г.

Авторы работы
Студенты группы КЭ-413

_____/ Б.А. Пащенко
_____/ А.В. Маслов
_____/ В.В. Романенко
« » _____ 2020 г.

Проект защищен с оценкой
_____/_____/
« » _____ 2020 г.

Челябинск, 2020

АННОТАЦИЯ

Пащенко Б.А., Маслов А.В., Романенко В.В. Разработка датчика бесконтактного измерения температуры с передачей параметров по беспроводному интерфейсу. – Челябинск: ЮУрГУ, ВШЭКН, КЭ-413; 52 с. 44 илл., библиогр. список – 5 наим.

В рамках курсовой работы было разработано устройство на основе отладочной платы XNUCLEO-F411RE, предназначенное для измерения температуры. В ходе разработки были выполнены следующие задачи:

1. Анализ требований к разработке.
2. Разработка общей и детальной архитектуры проекта.
3. Разработка кода проекта.
4. Оформление пояснительной записки к курсовому проекту.

Разработка программного обеспечения проводилась в двух средах разработки:

- StarUML – разработка архитектуры;
- IAR Embedded Workbench 8.40.2 – разработка кода на языке C++.

Пояснительная записка к курсовой работе оформлена в текстовом редакторе MS Word 2010.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6
1 АНАЛИЗ ТРЕБОВАНИЙ 7
1.1 Для разработки должна использоваться отладочная плата XNUCLEO-F411RE на базе микроконтроллера STM32F411RE 7
1.2 Архитектура должна быть представлена в виде UML диаграмм в пакете StarUML 9
1.3 Приложение должно быть написано на языке C++ с помощью компиллятора IAR Workbench for ARM 8.40.2 18
1.4 При разработке должна использоваться Операционная Система Реального Времени FreeRTOS 20
1.5 Передача значений по беспроводному интерфейсу должна осуществляться через BlueTooth Bee HC-06 22
1.6 Программное обеспечение должно измерять температуру 24
1.7 Период измерения температуры должен быть 100ms 25
1.8 К измеренной температуре должен быть применен цифровой фильтр 25
1.9 Для измерения температуры должен использоваться инфракрасный датчик температуры 25
1.10 Вывод значений должен производиться на экран с жидкими чернилами 26
1.11 Период вывода информации на индикатор должен быть 3 секунды 26
1.12 Формат вывода : «Температура:» XXX.XX 26
2 РАЗРАБОТКА АРХИТЕКТУРЫ ПРОЕКТА 27
2.1 Общая архитектура проекта 27
2.2 Разработка детальной архитектуры 28
2.2.1 Класс IGpio 28
2.2.2 Класс GpioPort 29
2.2.3 Класс Button 29
2.2.4 Класс ButtonTask 30
2.2.5 Класс SPI 30

2.2.6 Класс DisplayDriver 31
2.2.7 Класс IDisplayDriver 32
2.2.8 Класс EInkDisplay 32
2.2.9 Класс IDisplay 33
2.2.10 Класс DisplayView 34
2.2.11 Класс IDisplayView 34
2.2.12 Класс DisplayDirector 35
2.2.13 Класс Format 35
2.2.14 Класс USARTConfig 36
2.2.15 Класс USART 36
2.2.16 Класс BluetoothDriver 37
2.2.17 Класс IBluetoothDriver 38
2.2.18 Класс Bluetooth 38
2.2.19 Класс BluetoothDirector 39
2.2.20 Класс SMBus 40
2.2.21 Класс TemperatureSensor 40
2.2.22 Класс Filter 41
2.2.23 Класс IVariable 42
2.2.24 Класс Temperature 43
2.2.25 Класс IUnits 43
2.2.26 Класс Celsius, Kelvin, Fahrenheit 44
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 51
ПРИЛОЖЕНИЯ 52
ПРИЛОЖЕНИЕ А 52
ПРИЛОЖЕНИЕ Б ..... **Ошибка! Закладка не определена.**

ВВЕДЕНИЕ

Программное обеспечение (ПО) — наряду с аппаратными средствами, важнейшая составляющая информационных технологий, включающая компьютерные программы и данные, предназначенные для решения определённого круга задач и хранящиеся на машинных носителях.

ПО представляет собой алгоритм, реализованный в виде последовательности инструкций для процессора.

Для управления различными электронными устройствами необходим микроконтроллер.

Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи.

Температура является одной из наиболее часто измеряемых и контролируемых физических величин.

Измерения температуры широко используются и распространены в современном промышленном производстве.

Для контроля над тепловым режимом используются датчики температуры воздуха. Устройства для регулирования и контроля температуры это практичное решение, которое позволяет ощутить значительную экономию ресурсов для отопления и обогрева.

В настоящее время ощущается растущий интерес к стандартам и технологиям беспроводной связи.

Основным преимуществом передачи в беспроводной системе является то, что она может быть установлена эффективно, оперативно и без значительных затрат практически в любом месте.

Целью данной курсовой работы является разработка датчика бесконтактного измерения температуры с передачей параметров по беспроводному интерфейсу с питанием от солнечной батареи.

Курсовая работа выполнена в соответствии с СТО ЮУрГУ 04-2008.

1 АНАЛИЗ ТРЕБОВАНИЙ

1.1 Для разработки должна использоваться отладочная плата XNUCLEO-F411RE на базе микроконтроллера STM32F411RE

XNUCLEO-F411RE – отладочная плата компании Waveshare. В основе платы ARM Cortex-M4 микроконтроллер STM32F411RET6. Эта отладочная плата представляет собой гибкую платформу, позволяющую разработчикам реализовать собственные идеи и в кратчайшие сроки сделать прототип будущего изделия.

Разъемы ST Morpho платы XNUCLEO-F411RE обеспечивают полный доступ к линиям портов ввода/вывода (I/O) и дальнейшее периферийное расширение.

Поддержка mbed делает возможным быстрое построение прототипа устройства с использованием SDK и online инструментов. Комплексное бесплатное программное обеспечение (HAL библиотека) включает различные примеры софта. Изделие поставляется с отдельным модулем ST-Link/ V2.

Технические характеристики микроконтроллера STM32F411RET6:

- ядро: ARM 32-Бит Cortex-M4;
- рабочая частота: 100МГц;
- рабочее напряжение: 1.7…3.6В;
- память: 512кБ Flash, 128кБ SRAM;
- интерфейсы: 1 x SDIO, 1 x USB 2.0 FS, 5 x SPI or 5 x I2S, 3 x USART, 3 x I2C;
- АЦП/ЦАП: 1 x АЦП (12 Бит, 16 каналов).

Остальные технические характеристики:

- SPX3819M5: регулятор напряжения 3,3 В;
- AMS1117-5.0: регулятор напряжения 5,0 В;
- CP2102: преобразователь USB в UART;
- разъем Arduino: для подключения щитов Arduino;
- интерфейс ICSP: Arduino ICSP;
- USB TO UART: для отладки;
- разъем USB: интерфейс связи USB;

- интерфейс SWD: для программирования и отладки;
- заголовки ST Morpho: доступ к VCC, GND и всем входам / выходам, прост в расширении;
- 6-12 В постоянного тока;
- пользовательская кнопка;
- кнопка сброса;
- индикатор питания;
- пользовательский светодиод;
- 500 мА быстрый самовосстанавливающийся предохранитель;
- индикатор Rx / Tx последовательного порта;
- кристалл 8 МГц;
- кристалл 32,768 кГц.

Комплектация:

- 1 x Отладочная плата (XNUCLEO-F411RE);
- 1 x Программатор (ST-LINK/V2 (mini));
- 1 x Кабель (USB Type A Plug to Micro B Plug Cable);
- 1 x Кабель (USB Type A Plug to Receptacle Cable).

Технические параметры представлены в таблице 1.

Таблица 1 – Технические параметры

Серия оценочной/отладочной платы	xnucleo
Серия оценочной/отладочной платы	Cortex-M4
Разрядность шины данных, Бит	32
Наименование базового компонента	stm32f411ret6
Тип разъема для прямого подключения плат расширения	arduino
Наличие USB интерфейса	да

Окончание таблицы 1

Наличие установленного (в комплекте) дисплея	Нет
Наличие макетной области	Нет
Особенности	st-link/v2-mini, arduino-интерфейс
Вес, г	154

1.2 Архитектура должна быть представлена в виде UML диаграмм в пакете StarUML

StarUML – программный инструмент моделирования, который поддерживает UML (Унифицированный язык моделирования). StarUML ориентирован на UML версии 1.4 и поддерживает одиннадцать различных типов диаграмм, принятых в нотации UML 2.0. Он активно поддерживает подход MDA (Модельно-управляемая архитектура), реализуя концепцию профилей UML.

Model Driven Architecture (MDA) – новый подход к разработке ПО, предложенный консорциумом OMG. Архитектура MDA может предоставить ряд преимуществ по сравнению с существующими методиками: упрощение разработки многоплатформенных систем, простота смены технологической платформы, повышение скорости разработки и качества разрабатываемых программ и многое другое. Однако всё это станет возможным только тогда, когда среды разработки будут поддерживать новую технологию и полностью реализовывать её потенциал.

В основе MDA лежат понятия платформо-независимой и платформо-зависимой моделей (platform-independent and platform-specific models, PIMs and PSMs). В процессе разработки системы сначала создаётся PIM - модель, содержащая бизнес-логику системы без конкретных деталей её реализации, относящихся к какой-либо технологической платформе. Принципиальным является именно тот факт, что на этапе создания этой модели не принимается никаких решений по поводу её

реализации, разрабатываемый программный продукт не привязывается к технологиям.

Среда разработки StarUML настраивается в соответствии с требованиями пользователя и имеет высокую степень расширяемости, особенно в области своих функциональных возможностей.

StarUML предоставляет максимальную степень адаптации среды разработки пользователя, предлагая настройку параметров, которые могут влиять на методологию разработки программного обеспечения, проектную платформу и язык.

Главные особенности StarUML:

1. Точное соответствие стандарту UML:

StarUML строго придерживается спецификации UML, разработанной OMG для моделирования программ.

2. Открытый формат программной модели:

StarUML оперирует файлами в стандартном формате XML. Коды, написанные в легких для чтения структурах и форматах, могут быть легко изменены с помощью синтаксического анализатора XML.

3. Истинная поддержка MDA:

Можно создавать платформенно независимые модели, а платформенно зависимые модели (PSM) и исполняемые коды могут быть всегда автоматически сгенерированы на их основе.

Платформенно-зависимая модель PSM (Platform Specific Model) задает состав, структуру, функциональность системы применительно к конкретной платформе. Модель платформы задает технические характеристики, интерфейсы, функции платформы.

4. Применимость методологий и платформ:

Легко создаются не только модели под средства разработки для конкретных платформ типа .NET или J2EE, но также и для других основных структур программных моделей.

5. Превосходная расширяемость:

Любой язык, который поддерживает COM (модель для проектирования и создания компонентных объектов), может использоваться, чтобы вызывать StarUML или разрабатывать интегрированные дополнения. К примеру написать класс на языке программирования, который может поддерживать выполнение основных математических операций и превратить указанный класс в COM-объект, который можно будет использовать в любом языке программирования, поддерживающем COM-интерфейс.

6. Программная функция проверки модели:

Пользователи могут допускать ошибки в процессе моделирования. Чтобы предотвращать такие ситуации, StarUML автоматически проверяет модель программы.

7. Полезные дополнения:

StarUML включает много полезных дополнений с различными функциональными возможностями: генерация исходных текстов на языках программирования, конвертация исходных текстов в модели, импорт файлов Rational Rose, обмен модельной информацией с другими программными средствами, с использованием XMI, поддержка шаблонов проектирования.

XMI (XML Metadata Interchange) - это стандарт, позволяющий представить UML модель в виде XML документа. Он предназначен главным образом для хранения UML-данных, а также любых других данных.

Эти дополнения предоставляют дополнительные функции, увеличивают производительность, гибкость и функциональную совместимость моделей. [http://staruml.sourceforge.net/docs/user-guide(ru)/user-guide.pdf]

Моделирование с помощью UML осуществляется поэтапным построением ряда диаграмм, каждая из которых отражает какую-то часть или сторону системы либо ее замысла.

Модель – это абстракция, описывающая суть сложной проблемы или структуры без акцента на несущественных деталях, тем самым делая ее более понятной. Разработка программного обеспечения - не исключение. При построении сложной системы строятся ее абстрактные визуальные модели.

Диаграмма – это графическое представление множества элементов. Обычно диаграмма изображается в виде графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы подчиняются нотации UML и изображаются в соответствии с ней. Диаграмма классов может содержать не только классы, но также и интерфейсы, перечислимые типы, пакеты, различные отношения, инстанции и их связи.

Класс – это описание группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами и семантикой. Каждый класс является шаблоном для создания объекта. А каждый объект – это экземпляр класса. Диаграмма классов является частью логической модели системы и представляет статическую картину системы.

Для каждой системы строится не одна, а несколько диаграмм классов: возможно, что для каждого прецедента или сценария своя. На одних показывают подмножества классов, объединенные в пакеты, и отношения между ними, на других – отображают те же подмножества, но с атрибутами и операциями классов. Для представления системы разрабатывается столько диаграмм классов, сколько потребуется.

В нотации UML классы и объекты изображаются в виде прямоугольников (Рисунок 1).

Прямоугольник класса всегда делится на три секции (раздела), имя класса помещается в первую секцию. Во второй и третьей секциях могут указываться атрибуты и операции класса соответственно, эти секции могут быть пустыми и их можно скрыть. Класс должен описывать только одну сущность.

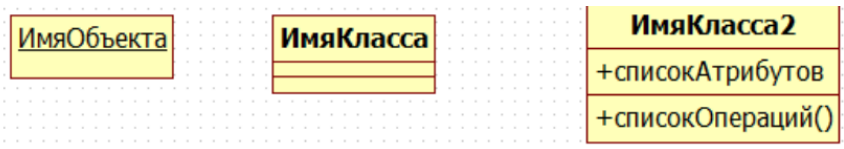

Рисунок 1 - Изображение классов и объектов

Видимость (visibility) – качественная характеристика описания свойств класса, характеризующая потенциальную возможность других объектов модели использовать это свойство (атрибут или операцию).

Видимость в языке UML может принимать одно из 4-х возможных значений и отображаться при помощи специальных символов:

• Открытый (public). Атрибут виден всем остальным классам. Любой класс, связанный с данным в рамках диаграммы или пакета, может просмотреть или изменить значение атрибута. Обозначается символом «+» перед именем атрибута.

• Защищенный (protected). Любой потомок данного класса может пользоваться его защищенными свойствами. Обозначается знаком «#» перед именем атрибута.

• Закрытый (private). Атрибут с этой областью видимости недоступен или не виден для всех классов без исключения. Обозначается знаком «-» перед именем атрибута.

• Пакетный (package). Атрибут является открытым, но только в пределах своего пакета. В StarUML данный атрибут обозначается значком «~».

Каждый класс должен обладать именем, отличающим его от других классов. Имя – это текстовая строка. Имя класса может состоять из любого числа букв, цифр и знаков препинания (за исключением двоеточия и точки) и может записываться в несколько строк.

На практике обычно используются краткие имена классов, каждое слово в имени которого традиционно пишут с заглавной буквы.

Для абстрактного класса имя класса записывается курсивом.

Атрибут (attribute) – содержательная характеристика класса, описывающая множество значений, которые могут принимать отдельные объекты этого класса.

Атрибут представляет некоторое свойство моделируемой сущности, которым обладают все объекты данного класса. Имя атрибута, как и имя класса, может представлять собой текст. На практике для именования атрибута используются одно или несколько коротких существительных, выражающих некое свойство класса, к которому относится атрибут.

Можно уточнить спецификацию атрибута, указав его тип, кратность (если атрибут представляет собой массив некоторых значений) и начальное значение по умолчанию.

Статические атрибуты класса обозначаются подчеркиванием. [https://prog-cpp.ru/uml-classes/]

Операция (Operation) – это спецификация или описание результата преобразования или запроса, который должен выполнить вызываемый объект.

Метод – это процедура, непосредственно реализующая операцию. Она имеет алгоритм и описание процедуры. Название метода совпадает с названием операции, перечень параметров может уточняться.

Графически операции представлены в нижнем блоке описания класса.

Допускается указание только имен операций. Имя операции, как и имя класса, должно представлять собой текст. На практике для именования операции используются короткие глагольные конструкции, описывающие некое поведение класса, которому принадлежит операция.

Можно специфицировать операцию, устанавливая ее сигнатуру, включающую имя, тип и значение по умолчанию всех параметров, а применительно к функциям – тип возвращаемого значения.

Классы могут находиться между собой в различных отношениях (связях). Базовыми отношениями в UML являются:

- зависимость;
- реализация;
- наследование;
- ассоциация;

- агрегация;
- композиция.

Зависимость – это такое отношение между элементами, при котором изменение одного элемента влечет за собой изменение другого элемента или требует представление ему дополнительной информации. В зависимость включаются все виды связей. Графически зависимость изображается в виде пунктирной стрелки, которая идет от зависимого класса к главному со стрелкой на конце (Рисунок 2). Зависимости применяются, чтобы показать, что один класс использует другой.

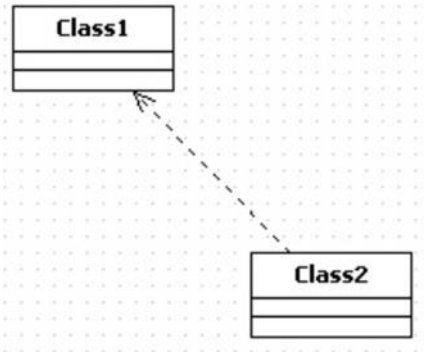

Рисунок 2- Зависимость между классами

Реализация определяет отношение между набором элементов, которые формируют спецификацию (клиент), и множеством элементов, которые формируют ее реализацию (поставщик). Обозначается пунктирной линией от поставщика к клиенту с не закрашенным треугольником на конце (Рисунок 3).

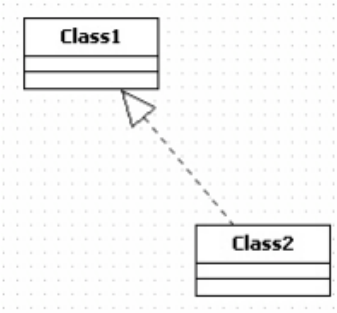

Рисунок 3 - Реализация

Наследование – таксономическое отношение между более общим элементом (родителем) и более специфичным элементом (потомком), который является полностью совместимым с первым элементом, но содержит дополнительную информацию. Оно используется для классов, пакетов, прецедентов и других элементов. Обозначается линией от потомка к родителю с не закрашенным треугольником на конце (Рисунок 4).

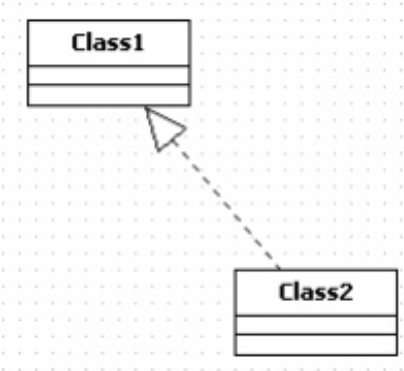

Рисунок 4 - Наследование

Ассоциация – отношение между двумя классификаторами (включая возможность ассоциации классификатора с самим собой). Обозначается линией (Рисунок 5).

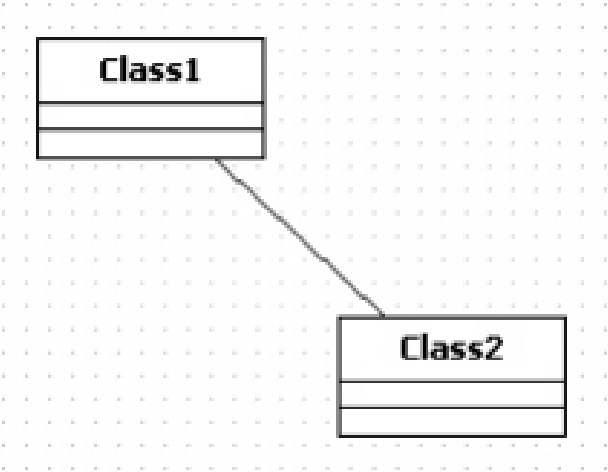

Рисунок 5 - Ассоциация

Агрегация – специфический тип ассоциации. На агрегат показывает незакрашенный ромбик в точке, где ассоциация соединяется с классификатором (конец ассоциации). Агрегация обозначает отношение «целое – часть». Классификатор, близ которого расположен полый ромбик, – целое (Рисунок 6).

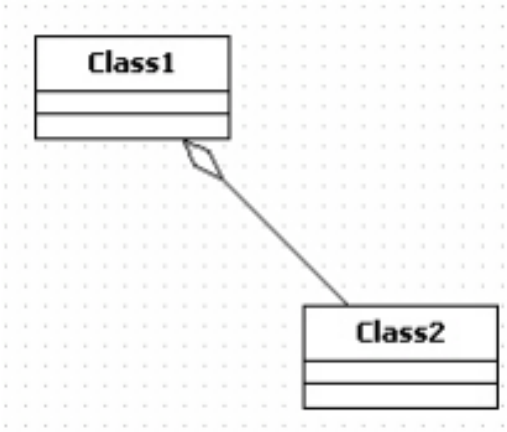

Рисунок 6 - Агрегация

Композиция – специфический тип ассоциации. Композиция обозначается заполненным ромбиком в точке, где ассоциация соединяется с классификатором (конец ассоциации). Композиция обозначает отношение целого и её неотделимой составной части, существование которой невозможно без целого. Классификатор, близ которого расположен заполненный ромбик, – целое (Рисунок 7).

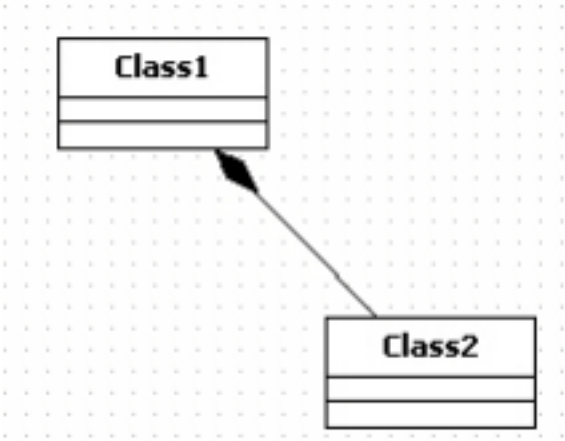

Рисунок 7 - Композиция

Для снижения сложности программы, ничего, кроме деления на части, пока не придумано. Правильная архитектура экономит очень много сил, времени и денег. Программу с хорошей архитектурой легче расширять и изменять, а также тестировать, отлаживать и понимать. Поэтому очень важно, чтобы программа не только хорошо работала, но и была хорошо организована. Для этого выбрали StarUML для разработки архитектуры программы.

1.3 Приложение должно быть написано на языке C++ с помощью компиллятора IAR Workbench for ARM 8.40.2

Си++ (англ. C++) – компилируемый строго типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования: процедурную, обобщённую, функциональную; наибольшее внимание уделено поддержке объектно-ориентированного программирования.

C++ – чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения.

Особенности C++:

- поддержка объектно-ориентированного программирования через классы (предоставляет все четыре возможности ООП – абстракцию, инкапсуляцию, наследование (в том числе и множественное) и полиморфизм);
- поддержка обобщенного программирования через шаблоны функций и классов;
- стандартная библиотека C++ состоит из стандартной библиотеки C (с некоторыми модификациями) и библиотеки шаблонов (Standard Template Library, STL), которая предоставляет обширный набор обобщенных контейнеров и алгоритмов;
- дополнительные типы данных;
- обработка исключений;
- виртуальные функции;

- пространства имён;
- встраиваемые (inline) функции;
- перегрузка (overloading) операторов;
- перегрузка имен функций;
- ссылки и операторы управления свободно распределяемой памятью.

IAR Embedded Workbench — комплексная среда разработки, многочисленные версии которой поддерживают большинство микроконтроллеров от различных производителей. Прежде всего, необходимо пояснить, что IAR EW является не просто программой, а именно комплексной средой разработки, поскольку состоит из целого комплекса программных и аппаратных инструментов. И хотя, как правило, аппаратные отладчики не поставляются в случае покупки только лицензии на пакет, так как в среду включена поддержка J TAG-адаптеров различных фирм, тем не менее, компания IAR рекомендует использовать в комплекте аппаратное обеспечение выпускаемое ею.

IAR Embedded Workbench предлагает обширную поддержку 8, 16 и 32-битных микроконтроллеров (MCU). Благодаря высокой скорости процессов оптимизации IAR Embedded Workbench позволяет быстро генерировать и выполнять необходимый код. Продукт IAR Embedded Workbench создан в сотрудничестве с ведущими мировыми IT-вендорами (Apple, Black&Decker, Cisco Systems, Ember, Ericsson, Hewlett-Packard, Motorola, Panasonic, Siemens и др.), что обеспечивает наиболее обширную поддержку архитектур процессоров на рынке.

По всему миру используется уже свыше 100 тысяч лицензий на ПО IAR Embedded Workbench. Многие успешные производители продуктов со встроенным программным обеспечением используют данное решение для критически важных приложений, в том числе в медицинской, промышленной и коммуникационной сфере. Надежность ПО IAR Embedded Workbench подтверждена многочисленными независимыми и коммерческими тестами.

Собственная технология IAR Systems позволяет в рамках одного инструментария легко перемещаться между 8, 16 и 32-битными микроконтроллерами с любой релевантной архитектурой, включая ядра ARM. Для стандартизации процесса разработки единый инструментарий также позволяет многократно использовать код в разных проектах. [https://store.softline.ru/iar/-29638/]

1.4 При разработке должна использоваться Операционная Система Реального Времени FreeRTOS

FreeRTOS – это операционная система реального времени с открытым исходным кодом для микроконтроллеров. Она упрощает программирование, развертывание, обеспечение безопасности, подключение и управление при работе с небольшими периферийными устройствами с малым энергопотреблением.

FreeRTOS распространяется бесплатно на условиях лицензии для продуктов с открытым исходным кодом. В состав операционной системы входят ядро и постоянно пополняемый набор библиотек программного обеспечения, которые можно использовать в различных секторах промышленности и областях применения. Ее можно использовать, например, для безопасного подключения небольших устройств с малым энергопотреблением к сервисам AWS Cloud либо более мощным периферийным устройствам. При разработке операционной системы FreeRTOS основное внимание было уделено надежности и простоте использования.

Микроконтроллеры содержат простой процессор с ограниченными ресурсами и используются во многих устройствах, включая бытовую технику, датчики, фитнес-трекеры, приборы промышленной автоматики и автомобили. Многие из этих устройств могут подключаться к облаку или локально к другим устройствам, но имеют ограниченные вычислительную мощность и объем памяти и обычно выполняют простые функциональные задачи. Микроконтроллеры часто работают под управлением операционных систем, у которых может не быть встроенных

функций для подключения к локальным сетям или облаку, из-за чего использованием таких устройств для Интернета вещей становится непростой задачей. FreeRTOS позволяет решить эту проблему и предоставляет как ядро, которое может работать на устройствах с малым энергопотреблением, так и библиотеки программного обеспечения, которые упрощают безопасное подключение к облаку (или другим периферийным устройствам). Таким образом, можно собирать данные с этих устройств Интернета вещей и выполнять требуемые действия.

Преимущества:

Открытый исходный код.

FreeRTOS выпускается на условиях лицензии для ПО с открытым исходным кодом. Это лицензия с малым количеством ограничений на повторное использование кода. Получить дополнительные сведения о сообществе разработчиков ПО с открытым исходным кодом FreeRTOS можно здесь.

Доверенное ядро.

Ведущие мировые компании доверяют ядру FreeRTOS, которое де-факто стало стандартом для микроконтроллеров и малых микропроцессоров, имеет проверенную надежность, небольшой объем и поддерживает широкий спектр устройств.

Сокращение времени вывода продуктов на рынок

FreeRTOS включает эталонные интеграции Интернета вещей, которые предварительно интегрированы с проектами FreeRTOS, портированными на микроконтроллерные оценочные платы (эти платы имеют сквозное подключение к облаку), и предварительно настроенные демонстрации, с помощью которых вы можете быстро приступить к работе над проектами. Вы можете мгновенно загружать код и компилировать его, чтобы уменьшить время вывода продуктов на рынок.

Безопасное программирование и развертывание устройств с малым энергопотреблением, а также подключение к ним и управление ими

В FreeRTOS имеется поддержка протокола TLS для безопасного подключения устройств к облаку. Вы можете без особого труда программировать часто используемые функции Интернета вещей на своих устройствах. Некоторые из доступных библиотек программного обеспечения позволяют настраивать устройства для работы в локальной сети с использованием стандартных вариантов подключения, например через Wi-Fi или Ethernet, или подключаться к мобильным устройствам с помощью технологии Bluetooth Low Energy *для передачи только коротких объёмов данных, когда это необходимо*. Кроме того, FreeRTOS включает библиотеку беспроводного обновления для удаленного обновления программного обеспечения устройств и добавления в него новых функций или исправлений системы безопасности, а также функцию подписания кода, которая позволяет проверить, не скомпрометирован ли код устройства в процессе развертывания и беспроводного обновления.

Широкая поддержка экосистемы

Экосистема наших партнеров предоставляет широкий выбор опций, в том числе сообщество, профессиональную поддержку, а также интегрированные средства для развертывания и повышения производительности труда. FreeRTOS предлагает гибкие возможности, позволяющие легко создавать решения Интернета вещей для разных наборов микросхем, и поддерживает более 40 архитектур.

1.5 Передача значений по беспроводному интерфейсу должна осуществляться через BlueTooth Bee HC-06

Bluetooth — это беспроводной интерфейс с небольшим радиусом действия, созданный в 1994 году инженерами шведской компании Ericsson. В 1998-м компании Ericsson, IBM, Intel, Nokia и Toshiba основали организацию Bluetooth Special Interest Group (Bluetooth SIG), которая и по сей день занимается разработкой и продвижением данной технологии.

Основными преимуществами Bluetooth по сравнению с конкурирующими решениями являются низкий уровень энергопотребления и невысокая стоимость

приемопередатчиков, что позволяет применять его даже в малогабаритных устройствах с миниатюрными элементами питания. Кроме того, производители оборудования не должны выплачивать лицензионные отчисления за использование интерфейса Bluetooth в своих изделиях. Разумеется, этот фактор также способствовал широкому распространению данного интерфейса.

Основным назначением Bluetooth является создание так называемых персональных сетей (Private Area Networks, PAN), которые обеспечивают возможность обмена данными между расположенными поблизости (например, внутри одного дома, помещения, транспортного средства) настольными и портативными ПК, периферийными и мобильными устройствами.

Bluetooth-модуль HC-06 простой способ беспроводного дистанционного управления вашим устройством.

Модуль Bluetooth производства ElecFreaks это простой в использовании модуль Bluetooth SPP, совместимый с существующими разъемами Xbee, выполненный на чипе CSR BC417, дает возможность взаимодействовать с другими устройствами по протоколу bluetooth 2.0.

Антенна уже встроена в плату модуля. Мощности передатчика хватает для работы на расстоянии до 30 метров при прямой видимости.

Сфера применения этого модуля не исчерпывается управлением. Его можно использовать и для пересылки показаний разнообразных сенсоров.

Рабочее напряжение этого bluetooth-модуля — 3,3 В, но его входы толерантны к 5 В.

Настройки по умолчанию представлены в таблице 2.

Таблица 2 – Настройки по умолчанию Bluetooth-модуля HC-06

Скорость передачи данных	9600 бод
Имя модуля	HC-06
Пароль для подключения	1234

Bluetooth-модуль HC-06 может выступать только в slave-режиме. Это означает, что он не может самостоятельно подключаться к другим Bluetooth-устройствам.

Характеристики Bluetooth-модуля HC-06 представлены в таблице 3.

Таблица 3 – Характеристики Bluetooth-модуля HC-06

Напряжение питания	3,3–6 В
Максимальное входное напряжение логической единицы	5 В
Выходное напряжение логической единицы	3,3 В
Максимальный ток потребления	45 мА
Скорость передачи данных	1200–1382400 бод
Дальность связи при прямой видимости	30 м

1.6 Программное обеспечение должно измерять температуру

Температура – физическая величина, характеризующая термодинамическую систему и количественно выражающая интуитивное понятие о различной степени нагретости тел.

Температура является одной из наиболее часто измеряемых и контролируемых физических величин.

Переход тепла от одного тела к другому указывает на зависимость температуры от количества внутренней энергии, носителями которой являются молекулы вещества. Согласно молекулярно-кинетической теории сообщаемая телу тепловая энергия, вызывающая повышение его температуры, преобразуется в энергию движения молекул.

Измерения температуры широко используются и распространены в современном промышленном производстве.

Для контроля над тепловым режимом используются датчики температуры воздуха. Устройства для регулирования и контроля температуры это практичное

решение, которое позволяет ощутить значительную экономию ресурсов для отопления и обогрева.

1.7 Период измерения температуры должен быть 100ms

Необходимо обеспечить периодический опрос канала АЦП к которому подключен датчик с периодичностью раз в 100ms и преобразование кода АЦП в значение температуры. Это можно реализовать с помощью операционной системы реального времени (ОСРВ).

1.8 К измеренной температуре должен быть применен цифровой фильтр

Требуемый вид фильтра:

$$T = \int \begin{pmatrix} 1 - e^{-\frac{dt}{RC}} & RC > 0\ sec, \\ 1 & RC \leq\ 0\ sec \end{pmatrix}$$

$$FilteredValue = OldFiltered + (Value - OldValue),$$

где dt – 100мс;

Value – текущее нефильтрованное измеренное значение температуры;

oldValue – предыдущее фильтрованное значение.

1.9 Для измерения температуры должен использоваться инфракрасный датчик температуры

Необходимо производить измерение температуры с датчика MLX90614.

Характеристики серии MLX90614:

точность в диапазоне до +50 0.5°C,

разрешение 0.02°C;

диапазон измерения -70°C … +380°C.

Мы использовали данный датчик, поскольку он совместим с выбранной платой, SMBus совместимый цифровой интерфейс и имеется в наличии.

1.10 Вывод значений должен производиться на экран с жидкими чернилами

Это модуль дисплея E-Ink, 4,2 дюйма, Разрешение 400x300, со встроенным контроллером, общается через интерфейс SPI.

Особенности:

- Рабочее напряжение: 3,3 В ~ 5 В
- Интерфейс: 3-проводной SPI, 4-проводной SPI
- Размер контура: 103,0 мм × 78,5 мм
- Размер дисплея: 84,8 мм × 63,6 мм
- Точка шаг: 0,212 × 0,212
- Разрешение: 400 × 300
- Цвет дисплея: красный, черный, белый
- Серый уровень: 2
- Время полного обновления: 15 с
- Мощность обновления: 26,4 МВт (typ.)
- Мощность в режиме ожидания: <0,017 МВт
- Угол обзора:> 170 °.

1.11 Период вывода информации на индикатор должен быть 3 секунды

Для обеспечения данной задержки используется операционная система реального времени (ОСРВ).

1.12 Формат вывода : «Температура:» XXX.XX

Значение температуры должно быть представлено численно.

2 РАЗРАБОТКА АРХИТЕКТУРЫ ПРОЕКТА

2.1 Общая архитектура проекта

Общая архитектура проекта, выполненная в программе StarUML, представлена на Рисунок 8 - Общая архитектура проекта.

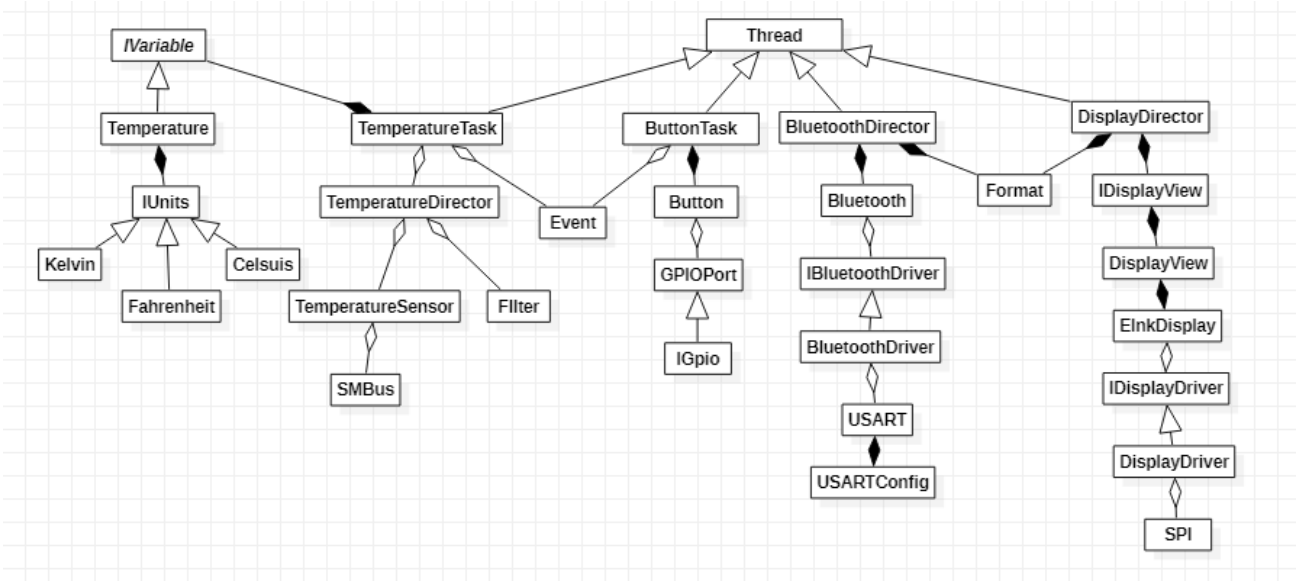

Рисунок 8 - Общая архитектура проекта

За аппаратную часть отвечают классы SMBus, SPI, USART.

За последовательный протокол обмена данными интерфейс отвечает класс SMBus.

За последовательный, периферийный интерфейс отвечает класс SPI.

Класс GPIOPort – отвечает за настройку универсальных входов и выходов.

Класс USARTConfig – отвечает за настройку параметров для класса USART.

Для работы с датчиком температуры используется класс TemperatureSensor. Код, получаемый с датчика передается в класс TemperatureDirector. Класс Filter – класс, необходимый для фильтрации получаемых значений.

Класс TemperatureTask – класс, ответственный за вызов метода для обновления измерений раз в 100мс, используя систему RTOS, за которую ответственный класс Thread.

Классы единиц измерения температуры объединены общим интерфейсом IUnits, в них значение которое получает класс Temperature, преобразуется в нужную величину.

Класс ButtonTask – класс, ответственный за опрос состояния кнопки, и по нажатию кнопки, посылает запрос на класс TemperatureTask по переключению единиц измерения.

Класс BluetoothDirector – класс, отвечающий за задачу передачи данных по Bluetooth раз в 1000мс. Bluetooth подключается с помощью USART.

Класс DisplayDirector – класс, ответственный за вывод показаний на e-paper дисплей, работу дисплея осуществляет класс DisplayDriver.

2.2 Разработка детальной архитектуры

2.2.1 Класс IGpio

Класс IGpio виртуальный класс, отвечающий за настройку портов (Рисунок 9)

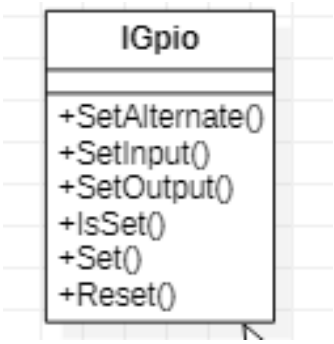

Рисунок 9 - Класс IGpio

У класса есть 6 публичных методов:

- SetInput() – режим входа.
- SetOutput() – режим выхода.
- SetAlternate() – альтернативный режим.
- IsSet() – состояние порта.
- Set() – установка 1.
- Reset() – установка 0.

2.2.2 Класс GpioPort

Отвечает за настройку портов, наследует все методы, которые есть у виртуального класса IGpio (Рисунок 10).

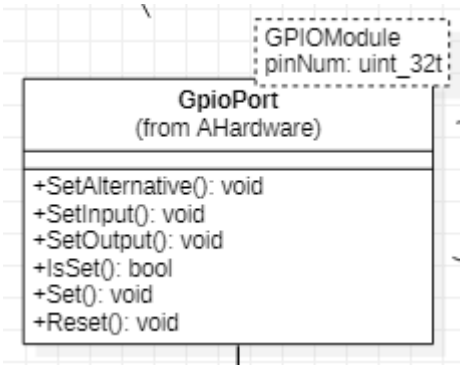

Рисунок 10 - Класс GpioPort

У класса есть 6 публичных методов:

- SetInput():void – режим входа.
- SetOutput(): void – режим выхода.
- SetAlternate(): void – альтернативный режим.
- IsSet(): bool – состояние порта.
- Set(): void – установка 1.
- Reset(): void – установка 0.

2.2.3 Класс Button

Отвечает за состояние кнопки (Рисунок 11).

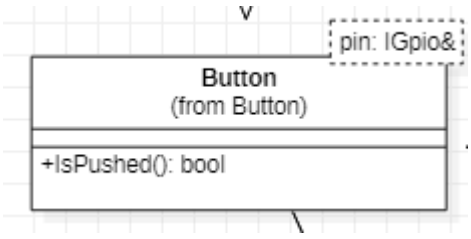

Рисунок 11 - Класс Button

Публичный метод:

IsPushed():bool – состояние кнопки (нажата/отжата)

2.2.4 Класс ButtonTask

Класс, отвечающий за опрос кнопки раз в 300мс (Рисунок 12).

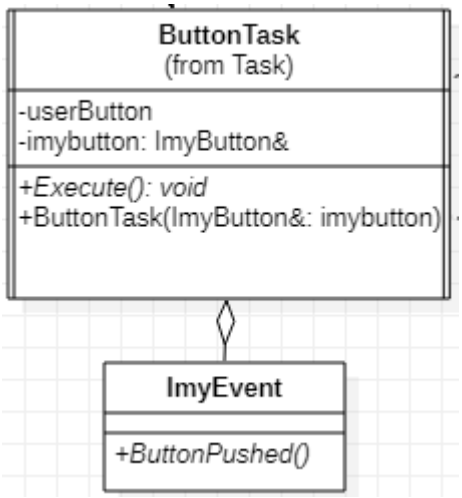

Рисунок 12 - Класс ButtonTask

Приватные атрибуты:

- userButton – объект класса PinsConfig, создающий пользовательскую кнопку.
- imybutton – ссылка на интерфейс ImyEvent.

Публичные методы:

- Execute():void – Метод, вызываемый ОСРВ, опрашивающий состояние кнопки
- ButtonTask(ImyButton&: imybutton) – Конструктор, инициализирующий ссылку.

2.2.5 Класс SPI

Отвечает за настройку периферийного интерфейса SPI (Рисунок 13).

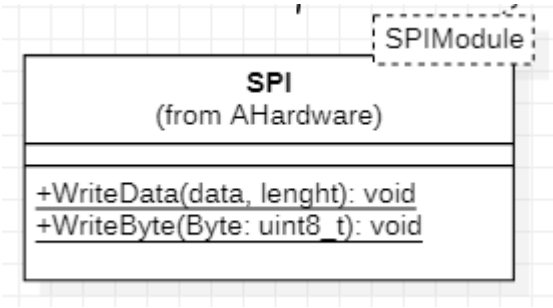

Рисунок 13 - Класс SPI

Публичные методы:

- WriteData(*data, lenght)): void – отправка данных.
- WriteByte(Byte: uint8_t): void – записать байт. Используется для заполнения массива данных байтами.

2.2.6 Класс DisplayDriver

Отвечает за работу и подключение дисплея (Рисунок 14).

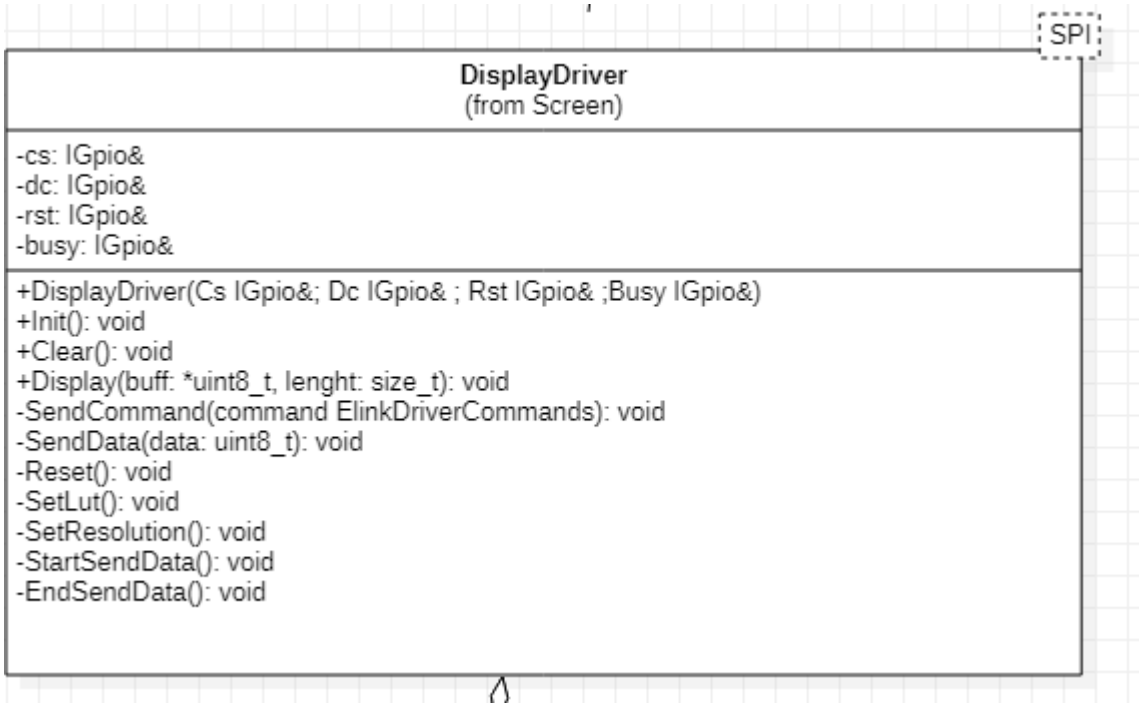

Рисунок 14 - Класс DisplayDriver

Приватные атрибуты:

Ссылки на класс IGpio для пинов cs,dc,rst,busy.

Публичные методы:

- DisplayDriver(Cs IGpio&, Dc IGpio&, Rst IGpio&, Busy IGpio&,) – конструктор, в котором происходит настройка режима пинов и настройка SPI.
- Init(): void – инициализация дисплея.
- Clear(): void – очистка дисплея.
- Display(buff: *uint8_t, length: size_t): void – отображение на дисплее.

Приватные методы:

- SendCommand(command ElinkDriverCommands): void – отправить команду.

- SendData(data: uint8_t): void – отправить данные.
- Reset(): void – сброс.
- Refresh(): void – обновить дисплей.
- SetResolution(): void – установить разрешение.
- StartSendData(): void – начать передачу данных.
- EndSendData(): void – закончить передачу данных.
- SetLut(): void – установка таблицы быстрого отображения.

2.2.7 Класс IDisplayDriver

Интерфейс IDisplayDriver отвечает за объединение всех драйверов общим интерфейсом (Рисунок 15).

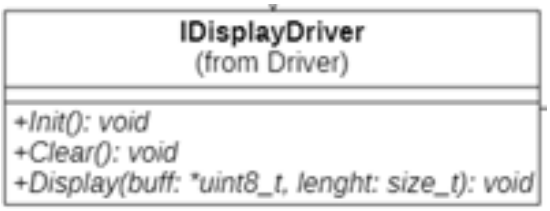

Рисунок 15 - Класс IDisplayDriver

Виртуальные методы:

- Init(): void – инициализация дисплея.
- Clear(): void – очистка дисплея.
- Display(buff: uint8_t, length, size_t): void – показ на дисплеее.

2.2.8 Класс EInkDisplay

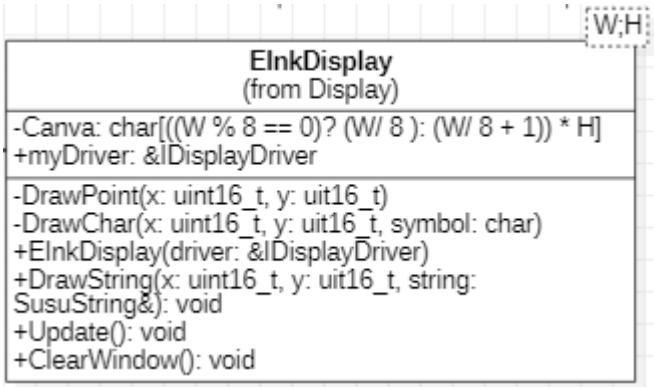

Рисунок 16 - Класс EInkDisplay

Класс отвечающий за взаимодействие с e-paper (Рисунок 16).

Публичный атрибут:

- myDriver: &IDisplayDriver – ссылка на интерфейс IDriver.

Приватные атрибуты:

- Canva: char[((W% 8 == 0)?(W/8)☹W/8+1))*H] –область вывода данных.

Публичные методы:

- EInkDisplay(driver: &IDisplayDriver) – конструктор, инициализирующий ссылку на IDisplayDriver, в теле которого осуществляется инициализация драйвера.
- DrawString(x: uint16_t, y: uit16_t, string: SusuString&): void – вывод строки на дисплей. Переопределяет виртуальный метод.
- Update(): void – обновление дисплея. Переопределяет виртуальный метод.
- ClearWindow(): void – очистка дисплея. Переопределяет виртуальный метод.
- DrawPoint(x: uint16_t, y: uit16_t) – метод рисования точки.
- DrawChar(x: uint16_t, y: uit16_t, symbol: char) – метод рисования символа.

2.2.9 Класс IDisplay

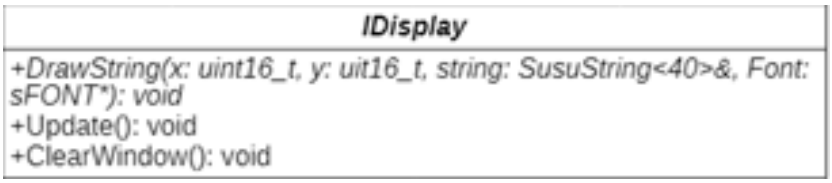

Рисунок 17 - Класс IDisplay

Интерфейс IDisplay отвечает за объединение дисплеев общим интерфейсом (Рисунок 17).

Виртуальные методы:

- DrawString(x: uint16_t, y: uit16_t, string: SusuString<40>&, Fony: sFONT*): void – вывод строки на дисплей.
- Update(): void – обновление дисплея.
- ClearWindow(): void – очистка дисплея.

2.2.10 Класс DisplayView

Класс отвечает за отображение на экране строки с выводом (Рисунок 18)

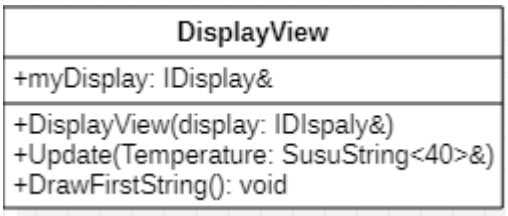

Рисунок 18 - Класс DisplayView

Приватный атрибут:

- myDisplay: IDisplay& – ссылка на интерфейс IDisplay.

Публичные методы:

- DisplayView(display: IDisplay&) – инициализирует ссылку на интерфейс IDisplay.
- Update(Temperature: SusuString<40>&) – обновляет дисплей и выводит на него значения
- DrawFirstString(): void – вывод первой строки

2.2.11 Класс IDisplayView

Виртуальный класс, отвечающий за показ значений на дисплее (Рисунок 19)

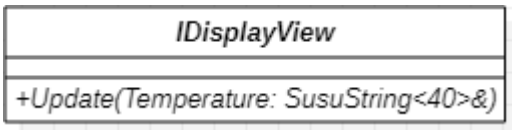

Рисунок 19 - Класс IDisplayView

- Update(Temperature: SusuString<40>&) – обновляет дисплей и выводит на него значения

2.2.12 Класс DisplayDirector

Класс, отвечающий за вывод показаний на дисплей раз в 3000мс, работает с ОСРВ (Рисунок 20).

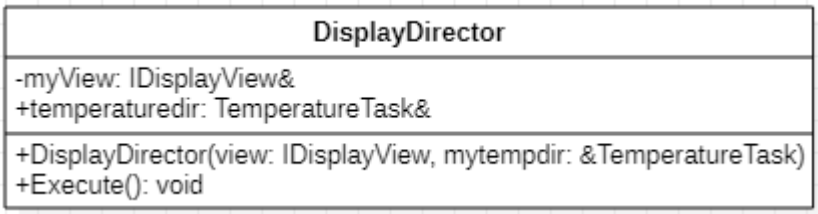

Рисунок 20 - Класс DisplayDirector

Приватные атрибуты:

- myView: IDisplayView& – ссылка на интерфейс IDisplayView.
- Temparaturedir: Temperature Task& – ссылка на класс TemparatureTask.

Публичные методы:

- DisplayDirector(view: IDisplayView, mytempdir: &TemperatureTask) – конструктор, инициализирующий ссылки на IDisplayView и TemparatureTask.
- Execute(): void – метод, вызываемый операционной системой реального времени.

2.2.13 Класс Format

Класс необходим для преобразования в нужный вид данных, получаемых от TemperatureTask, для вывода на дисплей и передачи через Bluetooth (Рисунок 21)

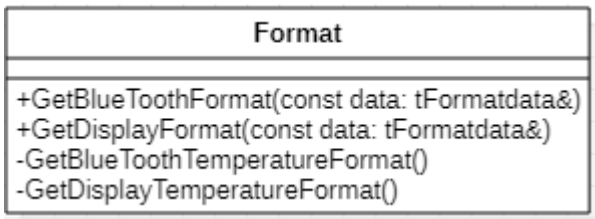

Рисунок 21 - Класс Format

Публичные методы:

- GetBlueToothFormat(const data: tFormatdata&) – возвращает данные для вывода по Bluetooth

- GetDisplayFormat(const data: tFormatdata&) – возвращает данные для вывода на дисплей строки со значениями температуры

Приватные методы:

- GetBlueToothTemperatureFormat() – преобразует данные температуры в строку для вывода по Bluetooth.
- GetDisplayTemperatureFormat() – преобразует данные температуры в строку для вывода на дисплей.

2.2.14 Класс USARTConfig

Класс, отвечающий за настройку USART, содержит классы enumeration, имеющие параметры для настройки (Рисунок 22)

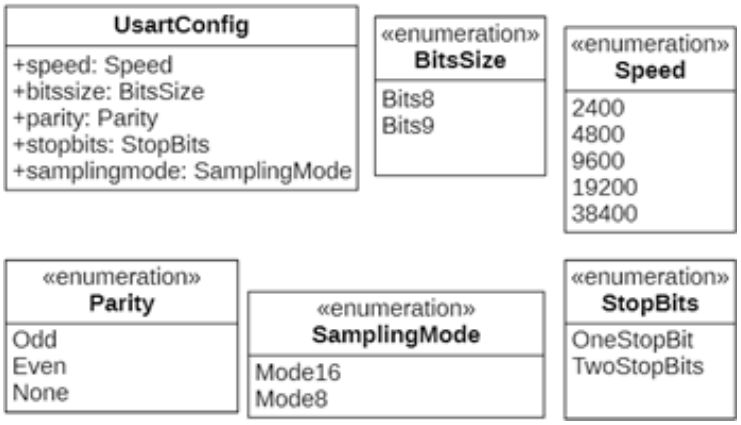

Рисунок 22 - Класс USARTConfig

2.2.15 Класс USART

Класс отвечает за работу с синхронно–асинхронным приемопередатчиком USART (Рисунок 23).

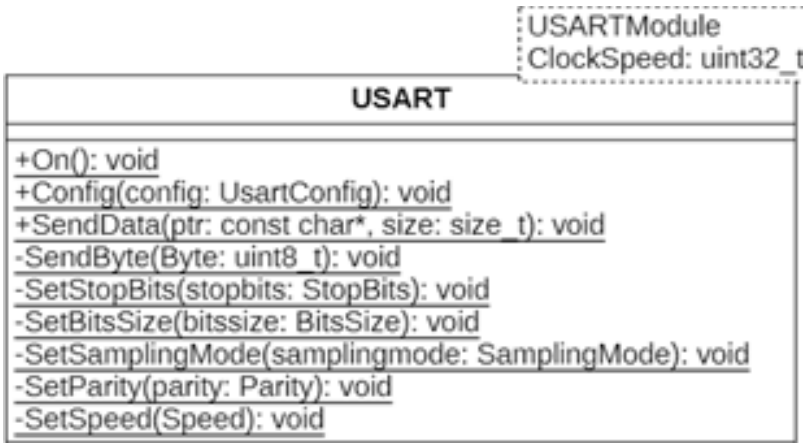

Рисунок 23 - Класс USART

Публичные методы:

- On()– включение модуля USART.
- Config() – первичная настройка USART.
- SendData()– отправить данные.

Приватные методы:

- SendByte() – отправить один бит данных.
- SetStopBits() – настройка количества стоп-битов
- SetBitsSize() – настройка количества битов данных для передачи
- SetSamplingMode() – настройка режима дискретизации
- SetParity() – настройка проверки четности для проверки передачи (режим none– для отключения проверки четности).
- SetSpeed() – настройка скорости приемопередатчика (по умолчанию настраиваем на 9600).

2.2.16 Класс BluetoothDriver

Класс, отвечающий за передачу настроек USART, и отправку данных (Рисунок 24 - Класс BluetoothDriver)

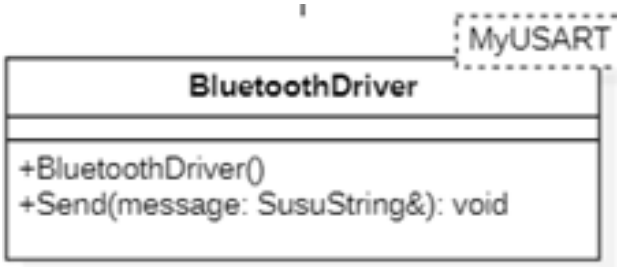

Рисунок 24 - Класс BluetoothDriver

- BluetoothDriver() – конструктор класса, в который передаются параметры настройки USART.
- Send() – отправка данных.

2.2.17 Класс IBluetoothDriver

Отвечает за интерфейс для работы с драйверами (Рисунок 25 - Класс IBluetoothDriver)

Рисунок 25 - Класс IBluetoothDriver

- Send() – отправить информацию

2.2.18 Класс Bluetooth

Класс отвечает за работу с конкретным Bluetooth-модулем (Рисунок 26).

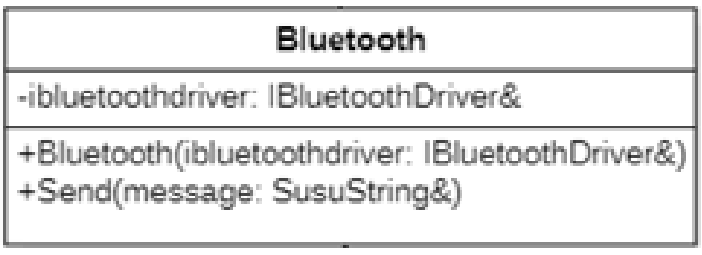

Рисунок 26 - Класс Bluetooth

Приватные атрибуты:

- ibluetoothdriver : IBluetoothDriver& – ссылка на интерфейс IBluetoothDriver.

Публичные методы:

- Bluetooth() – конструктор; служит для инициализации драйвера Bluetooth.
- Send() – Отправка данных через Bluetooth. Переопределяет виртуальный метод.

2.2.19 Класс BluetoothDirector

Класс отвечает за работу задачи Bluetooth (Рисунок 27).

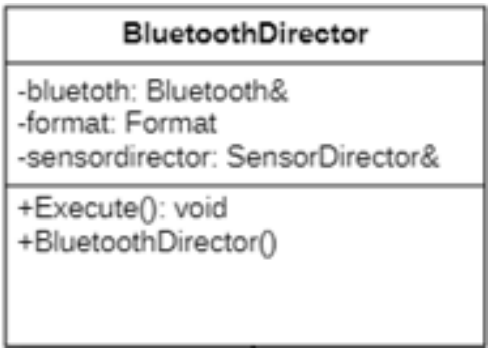

Рисунок 27- Класс BluetoothDirector

Приватные атрибуты:

- bluetooth : Bluetooth& – ссылка на класс Bluetooth.
- format : Format – создать объект класса Format.
- sensordirector : SensorDirector& – ссылка на класс SensorDirector.

Публичные методы:

- BluetoothDirector() – конструктор; инициализирует ссылки на классы SensorDirector и Bluetooth.
- Execute()– виртуальный метод операционной системы реального времени, определяемый в классе Thread, раз в секунду отправляет через Bluetooth.

2.2.20Класс SMBus

Отвечает за настройку интерфейса SMBus (Рисунок 28).

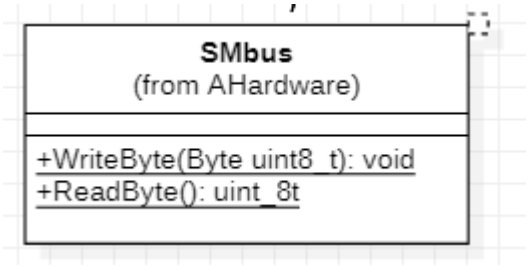

Рисунок 28 - Класс SMBus

Обладает публичными методами:

- WriteByte(Byte uint8_t): void – запись байта;
- ReadByte(): uint_8t – прочтение байта;
- Start() – начало работы, старт;
- Stop() – инициализация конца работы, стоп.

2.2.21 Класс TemperatureSensor

Класс, отвечающий за работу с инфракрасным датчиком MLX90614 (Рисунок 29).

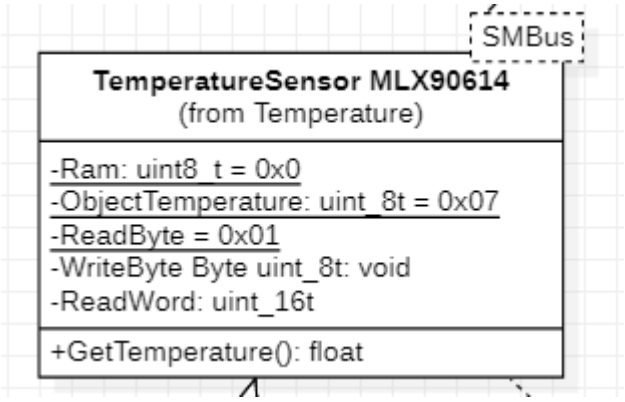

Рисунок 29 - Класс TempetureSensor

Приватные атрибуты:

- Ram: uint8_t=0x0 – команда памяти;
- ObjectTemperature: uint8_t=0x07 – команда измерения температуры объекта;
- ReadByte = 0x01 – команда прочтения байта;
- WriteByte Byte uint_8t: void– запись байта;

- ReadWord: uint_16t – прочтение слова;

Публичные методы:

- GetTemperature(): float – получение значения температуры.

2.2.22 Класс TemperatureDirector

Класс, отвечающий за получение данных с сенсора, передачи их в фильтр и получения оцифрованных значений (Рисунок 30).

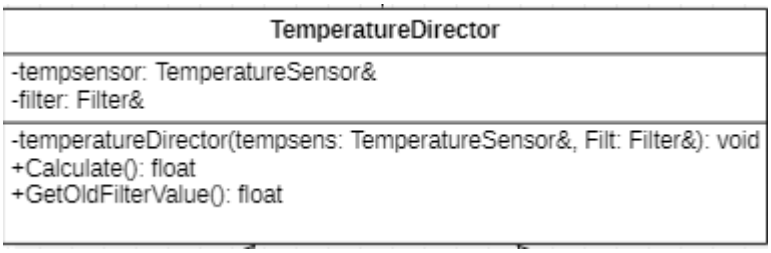

Рисунок 30 - Класс TemperatureDirector

Приватные атрибуты:

- tempsensor: TemperatureSensor& - Ссылка на класс TemperatureSensor
- filter: Filter& - Ссылка на класс Filter

Публичные методы:

- Calculate(): float – Расчет оцифрованных данных, полученных с фильтра
- GetOldFilterValue(): float – Хранение старых оцифрованных значений.

Приватный метод:

- temperaturedirector(): void – Конструктор, инициализирующий ссылки на классы TemperatureSensor и Filter

2.2.23 Класс Filter

Класс Filter используется для цифровой фильтрации значений температуры (Рисунок 31).

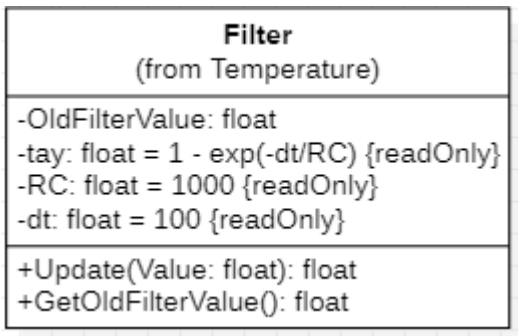

Рисунок 31 - Класс Filter

Приватные Атрибуты:

- OldFilterValue: float хранит предыдущее отфильтрованное значение влажности (изначально равняется нулю);
- tay: float = 1-exp(-dt/RC) {readOnly} – переменная, зависящая от RC и используемая для расчета отфильтрованного значения;
- dt: float = 100 {readOnly} шаг дискретизации, заданный в ТЗ равным 100мс, использующийся при расчете tay;
- RC: float = 1000 {readOnly} – постоянная времени.

Методы:

+Update(Value: float): float обрабатывает данные о температуре, полученные от TemperatureSensor, возвращает отфильтрованное значение и записывает текущее значение в OldFilteredValue.

GetOldFilterValue(): float – хранение старых оцифрованных значений

2.2.24 Класс IVariable

Рисунок 32 - Класс IVariable

IVariable – интерфейс переменных, объявляет виртуальные публичные методы (Рисунок 32):

- Get() – возвращение данных, состоящих из значения переменной и единицы измерения переменной.
- Calculate() – рассчитывает значение переменной.

2.2.25 Класс Temperature

Класс Temperature наследует методы интерфейса IVariables, а также имеет публичный метод (Рисунок 33).

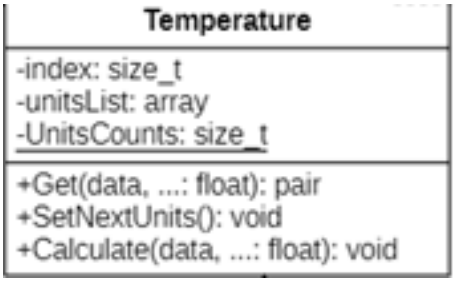

Рисунок 33 - Класс Temperature

Публичный метод:

- SetNextUnits() – по нажатию кнопки изменяет единицу измерения температуры на следующую (F→K→C).

Приватные атрибуты:

- index: size_t – счетчик нажатий кнопки.
- unitsList: array – массив единиц измерения.
- UnitsCounts: size_t – размер массива единиц измерения.

2.2.26 Класс IUnits

Ответственен за интерфейс единиц измерений, вызывает публичный метод (Рисунок 34).

Рисунок 34 - Класс IUnits

- GetTemperature() – рассчитывает значение температуры и возвращает данные, состоящие из значений температуры и единицы измерения.

2.2.27 Класс Celsius, Kelvin, Fahrenheit

Классы, которые служат для расчета каждой единицы измерения (Рисунок 35, Рисунок 36, Рисунок 37)

Рисунок 35 - Класс Celsius

Рисунок 36 - Класс Kelvin

Рисунок 37 - Класс Fahrenheit

2.2.28 Класс TemperatureTask

Класс отвечает за задачу температуры (Рисунок 38).

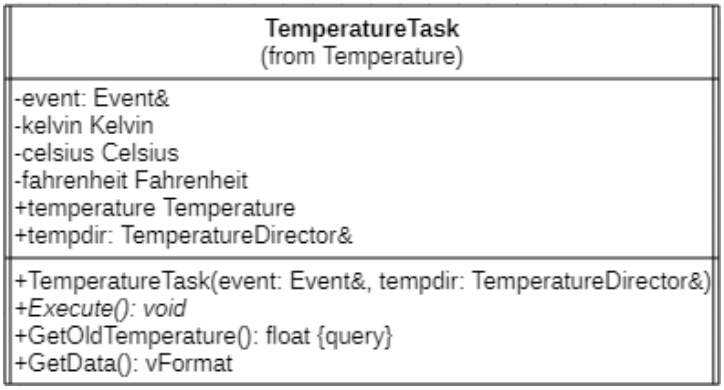

Рисунок 38 - Класс TemperatureTask

Приватные атрибуты:

- temperature – объект класса Temperature.
- event – ссылка на event.
- fahrenheit – объект класса Fahrenheit.
- kelvin – объект класса Kelvin.
- celsius – объект класса Celsius.
- tempdir – ссылка на класс TemperatureDirector.

Публичные методы:

- TemperatureTask() – конструктор, инициализирующий ссылку на класс TemperatureDirector и Event.
- Execute() – метод, вызываемый операционной системой реального времени, считывает данные с датчика раз в 100 мс.
- GetOldTemperature() – получение старой оцифрованной температуры
- GetData() – возвращает кортеж данных, состоящий из значений и единиц измерения температуры.

3 РЕЗУЛЬТАТ РАБОТЫ ПО

Для проверки работоспособности созданного программного обеспечения для инфрокрасного датчика температуры, а так же для проверки работы переключения единиц измерения температуры в последовательности (F-K-C), скачаем на android-смартфон программу Bluetooth Terminal HC-05 (Рисунок 39).

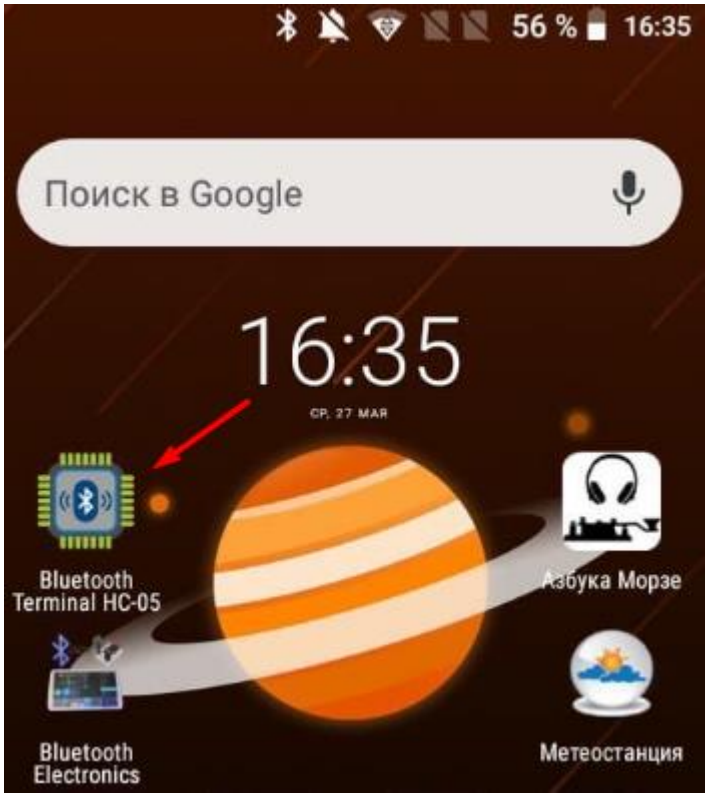

Рисунок 39 - Программа BluetoothTerminal

Следующим шагом нужно открыть программу, выбрать плату HC-06, подключиться к ней и ждать, когда будут приходить данные с датчика (Рисунок 40).

Для переключения единиц измерения нужно нажимать на кнопку USER на плате (Рисунок 41).

Рисунок 40 - Работа приложения

Рисунок 41 - Переключение единиц измерения

Проверим работоспособность датчика, померив температуру продукта из холодильника (Рисунок 42), температуру тела человека (Рисунок 43), температуру нагретого стакана с горячей водой().

Рисунок 42 - Температура продукта из холодильника

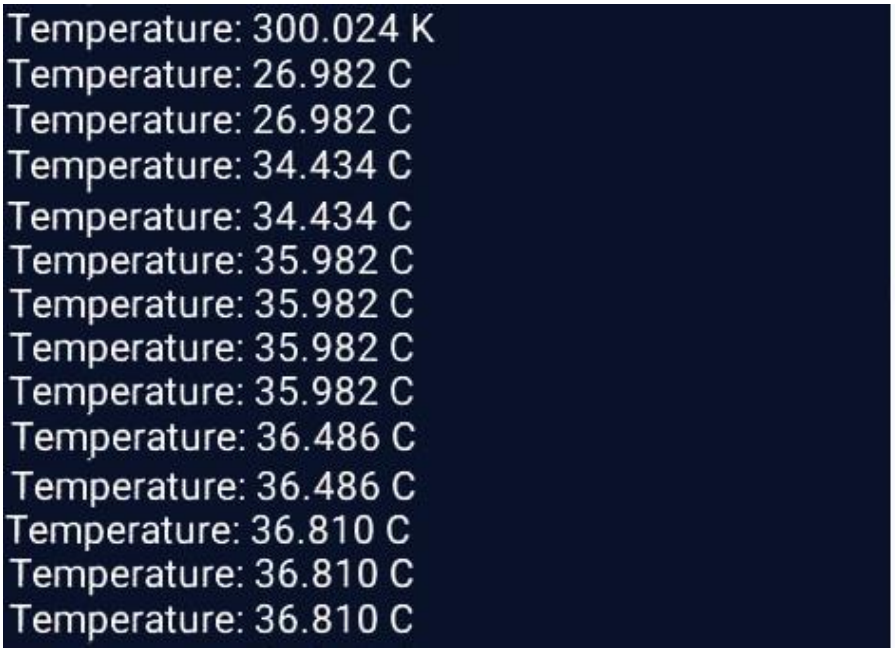

Рисунок 43 - Температура тела человека

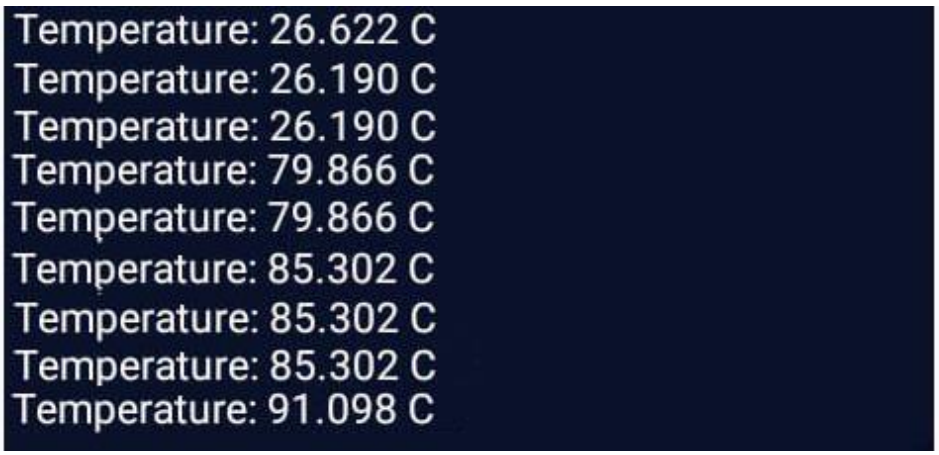

Рисунок 44 - Температура нагретого стакана с горячей водой

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано программное обеспечение для инфракрасного датчика температуры, реализовано оно на основе отладочной платы XNUCLEO–F411RE, в качестве инфракрасного датчика температуры был использован датчик MLX90614.

Вывод полученных данных реализован на дисплей с жидкими чернилами e-paper и выполнена передача данных по беспроводному протоколу Bluetooth с помощью модуля Bluetooth HC-06.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. СТО ЮУрГУ 04-2008 Стандарт организации. Курсовое и дипломное проектирование. – Челябинск: Изд-во ЮУрГУ, 2008. – 40 с.

2. https://www.waveshare.com/wiki/XNUCLEO-F411RE – XNUCLEO-F411RE.

3. http://staruml.io/ – Официальный сайт StarUML.

4. http://staruml.sourceforge.net/docs/user-guide(ru)/user-guide.pdf – Руководство пользователя StarUML.

5. https://m.habr.com/ru/post/420467/ – C++ обертка для «всех» Операционных Систем Реального Времени для CortexM4.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Код программы находится на веб-сервисе для хостинга IT-проектов GitHub – https://github.com/BogdanPashchenko/KURSOVOY