





Оглавление

Введение	7
1. Техническое задание.....	9
1.1 Назначение разработки и область применения	9
1.2 Технические требования.....	10
2. Анализ технического задания.....	11
2.1 Выбор платформы для реализации системы.....	11
2.2 Выбор датчиков.....	12
2.3 Выбор языка программирования.....	13
2.4 Выбор среды разработки	15
2.5 Выбор операционной системы.....	16
3. Разработка системы на структурном уровне.....	17
3.1 Структура аппаратно-программного комплекса.....	17
3.2 Разработка алгоритма работы системы.....	20
4. Программная реализация системы.....	21
4.1 Установка СУБД PostgreSQL 11	21
4.2 Настройка СУБД PostgreSQL 11	21
4.3 Реализация скрипта.....	23
4.5 Установка и настройка веб сервера с движком PHP.....	26
5. Тестирование системы.....	30
Заключение	35
Список литературы.....	36
Приложение 1.....	37
Приложение 2.....	45

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)			
Изм.	Лист	№ докум.	Подпись	Дата	Аппаратно-программный комплекс наблюдения за состоянием окружающей среды	Лит.	Лист	Листов
Разраб.		Лаптев А.С.		02.07				
Провер.		Гай В.Е.		02.07			6	45
Н. контр.		Гай В.Е.		02.07		НГТУ кафедра ВСТ		
Утверд.		Жевнерчук Д.В.		02.07				
					Пояснительная записка			

Введение

В настоящее время вопрос получения информации о состоянии окружающей среды поднимается всё чаще и чаще. Это бывает необходимо для того, чтобы автоматизировать те или иные процессы или предсказать состояние окружающей среды ближайшее будущее.

Прогнозирование погоды начинается с наблюдения за текущим состоянием атмосферы. Зная ее текущее состояние, синоптики могут затем прогнозировать предстоящие изменения погоды в ближайшие дни или недели. Первый в истории прогноз погоды, опубликованный в печати, был составлен именно Робертом Фицроем. Он был опубликован в английской газете Times 1 августа 1861 года. По одной из версий, именно неточность составляемых им прогнозов и стала причиной его добровольного ухода из жизни. Многочисленные погодные датчики, размещенные на поверхности Земли и над ней, в море и на орбите, измеряют целый ряд погодных параметров, которые помогают максимально нарисовать наиболее полную картину погоды на нашей планете. Сбор погодной информации ведется метеорологическими организациями по всему земному шару, а затем национальные метеослужбы обмениваются ею со своими коллегами в других странах. К основным погодным параметрам относятся: температура, атмосферное давление, влажность, скорость и направление ветра, осадки и их количество. Для их измерения на суше действует сеть метеостанций. В России таких метеостанций 1670, тогда как, например, в Китае их более 53 тысяч. Они могут обслуживаться как специалистами-метеорологами, так и быть полностью автоматизированными. В США, к примеру, действует сеть автоматизированных систем наблюдений (ASOS) за поверхностью. Такие метеостанции установлены в более чем 900 аэропортах по всей стране, где они собирают информацию о погодных явлениях.

А вот отслеживать в режиме реального времени местоположение и перемещение облачных образований, возникновение зон интенсивных осадков, фиксировать зоны опасных явлений, в том числе гроз, града, шквалов, следить за их развитием и перемещением помогают специальные погодные радары. В нашей стране разработкой и производством такого оборудования занимается концерн «Алмаз-Антей», известный своими системами противовоздушной и противоракетной обороны. Доплеровский метеорологический радиолокатор (ДМРЛ-С), разработанный этой ведущей

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						7
Изм.	Лист	№ докум.	Подп.	Дата		

оборонной корпорацией, относится к новому поколению радаров с двойной поляризацией сигнала. Современные радары ДМРЛ-С имеют радиус обзора 250–300 км и позволяют осуществлять циклические наблюдения с периодичностью от 3 до 15 минут в круглосуточном автоматизированном режиме. Они предоставляют данные с высоким пространственным разрешением (0,5–1 км) на площади до 200 тыс. км². Всего в планах Росгидромета до 2020 года значится установка около 140 радиолокаторов ДМРЛ-С. Специально разработанное для радиолокатора ДМРЛ-С программное обеспечение (ПО ВОИ «ГИМЕТ-2010») дает возможность соотносить метеоявления на карте ДМРЛ-С с синоптической ситуацией. Графическую информацию с таких радаров мы можем увидеть на картах осадков, имеющих на многих погодных сайтах.

В США также существует сеть метеорадаров, которая включает более чем 120 доплеровских радаров. Недавно они были усовершенствованы с помощью технологии Dual Polarization Technology, аналогичной той, что применили в ДМРЛ-С. На данный момент сеть погодных радаров в США считается самой развитой в мире. Радарами покрыта практически вся территория, причем восточная часть страны с большим запасом. Именно поэтому краткосрочный прогноз погоды в Вашингтоне и Нью-Йорке считается одним из самых точных на планете. В России сейчас также реализуется программа развития радиолокационной сети, новые радары строятся, прежде всего, в Центральном регионе, на юге Сибири и Дальнего Востока.

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		8

1. Техническое задание

1.1 Назначение разработки и область применения

Разрабатываемая система предназначен для наблюдения за изменениями окружающей среды и интеграции с сервисом «Народный мониторинг».

Сбор данных о погоде осуществляется непосредственно с помощью датчиков и обрабатывается программно-аппаратным модулем на основе мини-эвм. Дополнительными функциями являются: сохранение данных в базу данных на основе СУБД PostgreSQL и визуализация данных на диаграмме.

Целью проекта аппаратно-программной системы наблюдения за состоянием окружающей среды является создание прототипа системы, позволяющей получать точные данные об окружающей среде в режиме реального времени, а также возможность интеграции с системами умного дома и проектов IoT в целом.

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						9
Изм	Лист	№ докум.	Подп.	Дата		

1.2 Технические требования

Так как разрабатываемая система предназначена для наблюдения за изменениями погодных условий, сохранения данных и отправки данных на сторонние серверы, то выбор аппаратно-технических средств будет следующим:

1. Аппаратные средства:
 - a. Одноплатный компьютер Raspberry Pi 3;
 - b. Датчик BME280;
 - c. Адаптер питания Apple Model A1400.
 - d. USB модем Yota 4G LTE
2. Программное обеспечение:
 - a. Raspberry Pi OS;
 - b. Python 3;
 - c. PostgreSQL 11;
 - d. HTML, CSS, JavaScript, jQuery, PHP7.

Разрабатываемая система должна обладать следующими функциями:

- Получать показания датчиков влажности, температуры и давления;
- Отправлять показания на сервис «Народный мониторинг»;
- Дублировать данные в базу данных PostgreSQL.

2. Анализ технического задания

2.1 Выбор платформы для реализации системы

Рассмотрим основные аппаратные модули используемые для реализации системы:

1. Мини ЭВМ Raspberry Pi 3.

Выбранный микрокомпьютер обладает набором свойств, который позволяет решить поставленную задачу максимально комфортными способами. Так как этот компьютер разработан на основе архитектуры ARM и разработчиком этой аппаратной платформы предоставляется программное обеспечение, в виде операционной системы адаптированной под данный микропроцессор, мы можем использовать его в довольно широком спектре задач. На этом компьютере может быть установлена операционная система Raspberry Pi OS которая относится к семейству Linux систем, что предоставляет нам очень много способов решения поставленной и вероятных задач. С помощью Raspberry Pi можно осуществить не только сбор и передачу показаний датчика, а так же их хранение, обработку и визуализацию. Данная мини ЭВМ имеет размер примерно как у банковской пластиковой карточки и обладает весьма хорошим соотношением энергопотребления и предоставляемых ресурсов.

2. Датчик BME280.

В поисках оптимального датчика, было опробовано достаточное количество сенсоров, используемых для измерения основных метрик погоды, таких как температура, давление и влажность. Зачастую такие датчики имеют относительно большой размер и низкую точность показаний, а также довольно высокую стоимость. Датчик BME280 включает в себя сразу три сенсора: температура, влажность и давление. Данный выбор является самым оптимальным для выполнения поставленной задачи, так как имеет небольшой размер и энергопотребление.

2.2 Выбор датчиков

В настоящее время на рынке радиокомпонентов довольно большой выбор разного типа сенсоров, но каждый датчик следует использовать в соответствующей ситуации.

Так как необходимо получать данные о погоде по трём параметрам, то придётся использовать термометр, гигрометр и барометр для сбора данных.

Для того, чтобы сделать всю систему максимально компактной, будем искать датчик с оптимальными габаритами и энергопотреблением.

Таблица сравнения некоторых датчиков, представленных на российском рынке:

Таблица 1 - Сравнение датчиков

	ВМЕ280(термометр)	ВМЕ280(гигрометр)	ВМЕ280(барометр)	DS18B20	HDC1008YPAT	BMP180
Диапазон значений	-40...85°C	0...100%	300...1100 гПа	-55...125°C	0...100%	30...1100кПа
Точность (±)	0.3%	0.25%	11.5 Па/К	0.3%	4%	11.5 Па/К
Напряжение питания (В)	3.6			5.5	2.7	1.8
Максимальное энергопотребление (мкА)	3.6			1500	1.2	5
Габариты (ДхШхВ мм ³)	2.5 x 2.5 x 0.93			3 x 3 x 0.85	2.07 x 1.62 x 0.41	3.6 x 3.8 x 0.93
Вес (г)	0.04			0.3	0.254	0.04
Рыночная цена (руб.)	310			180	130	103

Из таблицы видно, что относительно оптимальным вариантом является выбор датчика ВМЕ280 так как он имеет в себе целых три встроенных сенсора при довольно малых габаритах, низком энергопотреблении и цене.

2.3 Выбор языка программирования

В процессе поиска языка программирования был проведён поиск самых популярных языков программирования на начало 2020го года. На просторах интернета была найдена статья с подробным разбором анализа данных, полученных от сервиса Google Trends и в данной статье фигурирует следующая таблица: [ссылка](#)

Rank	Change	Language	Share	Trend
1		Python	29.72 %	+4.3 %
2		Java	19.03 %	-1.9 %
3		Javascript	8.2 %	+0.1 %
4		C#	7.28 %	-0.2 %

Рисунок 1 - Сравнение популярности различных языков программирования

Глядя на таблицу, можно увидеть, что самым популярным языком программирования на начало 2020го года является Python. Так же, данный язык программирования имеет низкий порог вхождения и является одним из самых быстрых.

Python является высокоуровневым языком программирования общего назначения который ориентирован на повышение производительности разработчика и читаемости кода. Синтаксис языка является одним из самых читаемых и минималистичных языком программирования на данный момент.

Этот язык поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование, что позволяет писать довольно гибкий, удобочитаемый и поддерживаемый код. Неотъемлемой частью этого ЯП являются динамическая типизация, автоматическое управление памятью, механизмы обработки исключения, высокоуровневые структуры данных, а также поддержка многопоточных вычислений. Все эти плюсы предоставляют возможность писать код максимально быстро и комфортно.

Помимо перечисленных преимуществ, Python является кроссплатформенным языком и позволяет создавать кроссплатформенные

приложения, которые могут исполняться как на Linux так и на Windows
подобных системах.

					ВКР-НГТУ-09.03.01-(15-ВМ-6)-011-2020 (ПЗ)	Лист
						14
Изм	Лист	№ докум.	Подп.	Дата		

2.4 Выбор среды разработки

Для того, чтобы писать код приложения, необходим текстовый редактор, компилятор и среда выполнения, но если использовать текстовый редактор без каких-либо дополнений, то скорость и комфортность написания кода значительно снижается. Для того, чтобы повысить комфортность работы, необходимо использовать среду разработки (сокр. IDE).

В настоящее время, на рынке программного обеспечения, существует множество разнообразных вариантов сред разработки, в них входят как платные, так и бесплатные экземпляры. Рассмотрим некоторые из них:

- Eclipse IDE – платформа общего назначения, предназначенная для разработки программного обеспечения на нескольких языках. Это среда разработки кастомизируется с помощью плагинов. Для того, чтобы получить возможность работы со средствами языка python в этой среде разработки, необходимо установить дополнение, которое предоставит разного рода инструменты в виде подсветки синтаксиса, автодополнения и проверки кода. Данная среда разработки является открытой и распространяется бесплатно. Также, это программное обеспечение является кроссплатформенным.
- PyCharm – эта среда разработки разрабатывается компанией JetBrains. Данная IDE имеет ряд преимуществ по сравнению со многими: довольно приятный дизайн, богатый набор инструментов, а также два возможных варианта получения: платный и бесплатный. В случае если необходимы примитивные средства работы с языком такие как: редактор кода, отладчик, инструменты рефакторинга, инспекция кода и поддержка системы контроля версий, то можно использовать бесплатную версию, называемую «Community Edition». Если этого функционала недостаточно, то есть вариант с более обширным набором инструментов таких как: поддержка WEB фреймворков, профайлер, удалённая разработка, работа с базами данных и так далее, то эти возможности предоставляет платная версия ПО, называемая «Professional Edition».

В процессе анализа было принято решение, что для разработки будет использована среда разработки PyCharm от JetBrains, так как нет необходимости выполнять дополнительные действия для начала работы.

2.5 Выбор операционной системы

Аппаратно-программная система наблюдения за изменениями погодных условий, не требует каких-либо сложных вычислений и должна работать на мини-эвм Raspberry Pi 3. Отсюда следует, что нам необходимо использовать ПО, предоставляемое самим разработчиком аппаратной платформы, которое поставляется бесплатно и его можно скачать с официального сайта напрямую. К выбранной мини-эвм предоставляется программное обеспечение в виде операционной системы Raspberry Pi OS (в прошлом Raspbian). Эта операционная система является системой семейства Linux.

В данной операционной системе уже реализованы модули ядра, отвечающие за работу с колодкой портов ввода-вывода, расположенной на печатной плате мини-эвм, что сокращает объём работ и предоставляет удобный интерфейс для оперирования данными.

Средства операционной системы позволяют с лёгкостью устанавливать необходимые пакеты программного обеспечения для реализации поставленных задач.

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						16
Изм	Лист	№ докум.	Подп.	Дата		

3. Разработка системы на структурном уровне

Для того, чтобы упростить понимание принципа работы системы, необходимо составить структурную схему аппаратно-программного комплекса. Это позволит сформировать представление о том как связаны между собой аппаратные и программные компоненты.

3.1 Структура аппаратно-программного комплекса

Система должна состоять из следующих компонент:

- Raspberry Pi для агрегации данных и их распределения на различные системы хранения или отображения.
- Датчик для сбора показаний о состоянии окружающей среды;
- База данных для хранения статистики;
- Сторонний сервис для отображения показаний.

После сбора данных об имеющихся компонентах, можно составить следующую диаграмму:

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						17
Изм	Лист	№ докум.	Подп.	Дата		

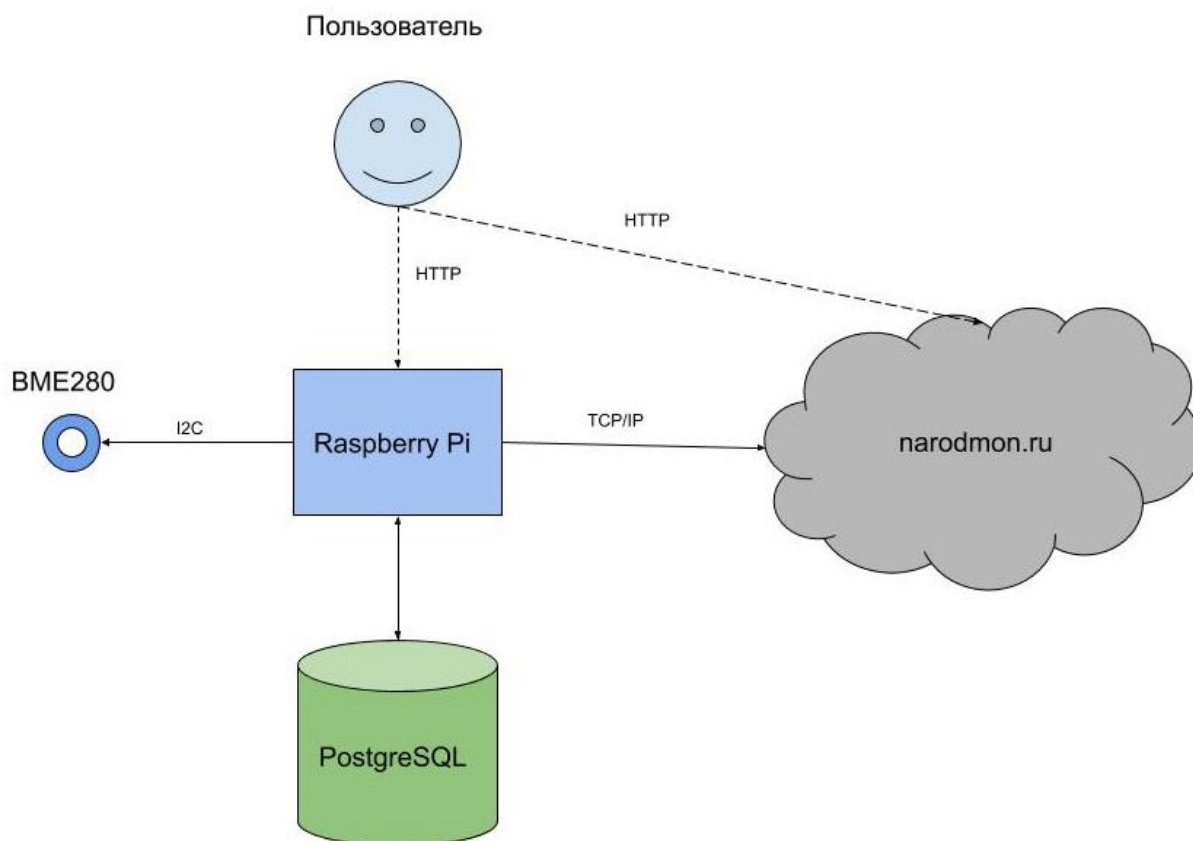


Рисунок 2 - Структура системы

На данной диаграмме показаны компоненты системы, которые взаимодействуют друг с другом определённым образом.

Рассмотрим взаимодействие компонент более подробно.

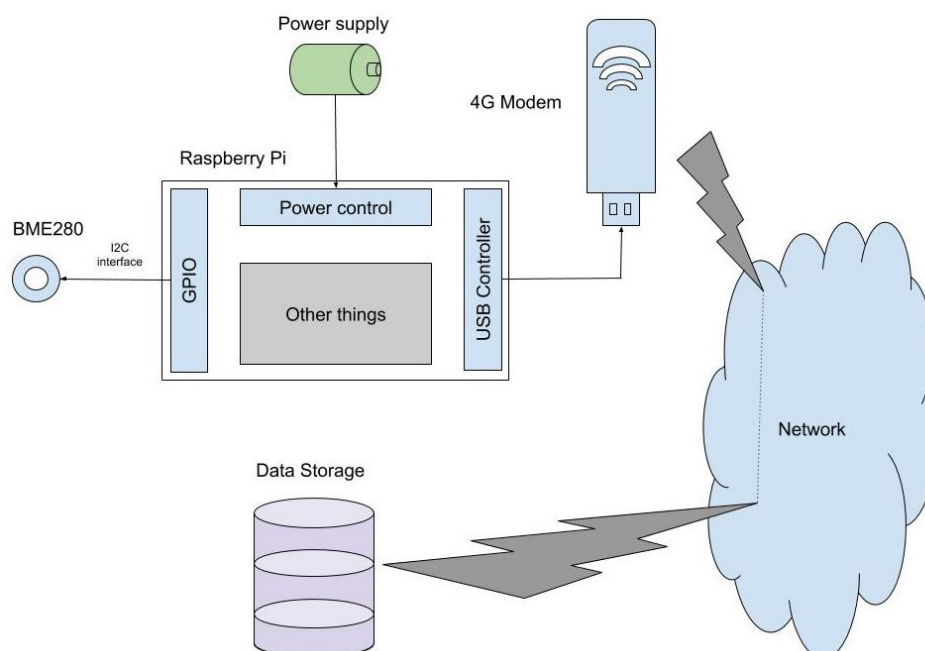


Рисунок 3 - Структурная диаграмма компонент системы

На рисунке 3 можно увидеть схему взаимодействия компонент системы на структурном уровне. Это позволит увидеть больше деталей процесса взаимодействия.

Из диаграммы выше понятно, что система делится на несколько компонент:

- Датчик
- Микрокомпьютер
- Источник питания
- Адаптер для связи с сетью
- Хранилище данных
- Сеть, к которой подключены микрокомпьютер и хранилище данных.

1.2 Разработка алгоритма работы системы

Разрабатываемый аппаратно-программный комплекс должен работать по следующему алгоритму:



Рисунок 4 - Алгоритм работы системы

В начале работы программы происходит опрос сенсоров. Затем скрипт отправляет данные на сервис «Народный мониторинг» и сохраняет данные в базу данных. Этот алгоритм должен повторяться каждые 5 минут.

Повтор каждые пять минут может осуществляться с помощью разных инструментов, начиная от планировщиков написанных для Python, заканчивая встроенным планировщиком задач для Linux, CRON.

4. Программная реализация системы

4.1 Установка СУБД PostgreSQL 11

В качестве машины, выполняющей роль сервера баз данных, была выбрана такая же мини-эвм, как и для системы мониторинга. На эту мини-эвм была предустановлена операционная система Raspberry Pi OS.

Для установки СУБД PostgreSQL 11 на Raspberry Pi, необходимо предварительно обновить список пакетов командой *sudo apt-get update*. После выполнения этой команды будет произведён запрос данных от репозиториев и локальный список пакетов будет обновлён.

После обновления можно приступить непосредственно к установке необходимого программного обеспечения. Так как в репозиториях для выбранной операционной системе присутствуют установочные пакеты postgres, можно установить всё необходимое программное обеспечение одной командой: *sudo apt-get install postgresql postgresql-contrib*. После начала выполнения установки установщик пакетов соберёт информацию о зависимостях и предоставит данные о том, сколько памяти будет использовано после установки. На данном этапе установка будет ожидать ответа от пользователя. Если дать положительный ответ, то через несколько минут всё необходимое будет установлено.

4.2 Настройка СУБД PostgreSQL 11

Для начала работы с базой данных Postgres, её необходимо настроить.


```

pi@postgres: ~
GNU nano 3.2 /etc/postgresql/11/main/pg_hba.conf

# TYPE      DATABASE      USER      ADDRESS      METHOD
# "local" is for Unix domain socket connections only
local      all             all                                     peer
# IPv4 local connections:
host       all             all        127.0.0.1/32  md5
host       all             all        0.0.0.0/0     md5
# IPv6 local connections:
host       all             all        ::1/128      md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local      replication  all                                     peer
host       replication  all        127.0.0.1/32  md5
host       replication  all        ::1/128      md5

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^_ Replace      ^U Uncut Text   ^T To Spell

```

Рисунок 5 - pg_hba.conf

На скриншоте выше, продемонстрирован конфигурационный файл, в котором разрешены внешние подключения. По умолчанию все внешние подключения игнорируются.

После разрешения внешних подключений необходимо указать некоторые настройки соединения.

```

pi@postgres: ~
GNU nano 3.2 /etc/postgresql/11/main/postgresql.conf

#-----
# - Connection Settings -

listen_addresses = '*'           # what IP address(es) to listen on;
                                  # comma-separated list of addresses;
                                  # defaults to 'localhost'; use '*' for all
                                  # (change requires restart)
port = 5432                      # (change requires restart)
max_connections = 100            # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                  # (change requires restart)
#unix_socket_group = ''          # (change requires restart)
#unix_socket_permissions = 0777 # begin with 0 to use octal notation

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^_ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line

```

Рисунок 6 - postgresql.conf

На скриншоте выше продемонстрирован участок конфигурационного файла, в котором указаны адреса с которых можно принимать соединения, норма порта на котором сервер будет принимать соединения и максимальное число одновременных подключений.

После выполнения всех настроек необходимо запустить сервер командой *sudo service postgresql start*.

4.3 Реализация скрипта

Программная реализация системы заключается в написании скрипта на языке Python 3, который будет запускаться раз в пять минут, собирать данные с датчиков, сохранять данные в базе данных и отправлять на сервис «народный мониторинг».

На первых строках после объявления зависимостей, можно увидеть две строки, которые служат для конфигурации I2C интерфейса на колодке ввода-вывода Raspberry Pi 3.

```
11 port = 1
12 bus = smbus2.SMBus(port)
```

Рисунок 7 - Объявление порта ввода-вывода

Из скриншота выше можно понять, что используется порт под номером 1 и создаётся экземпляр класса порта.

После конфигурации порта, необходимо объявить переменные, содержащие в себе некоторые данные для составления пакета данных для отправки на сервис «народный мониторинг».

```
18 DEVICE_MAC = 'B8:27:EB:A9:3A:C5'
19
20 SENSOR_ID_1 = 'T1'
21 SENSOR_ID_2 = 'P1'
22 SENSOR_ID_3 = 'H1'
```

Рисунок 8 - Объявление переменных для "Народного мониторинга"

Переменная «DEVICE_MAC» служит для идентификации устройства в системе агрегатора. Переменные «SENSOR_ID_1», «SENSOR_ID_2», «SENSOR_ID_3» содержат в себе имена датчиков. Эти имена служат для создания пакета данных, отправляемого на сервер «народного мониторинга». Сервис распознаёт датчики именно по этим именам для дальнейшей обработки.

Основная функция скрипта состоит из нескольких частей:

```
87     bme280.load_calibration_params( bus, bme280_address )
88     bme280.data = bme280.sample( bus, bme280_address )
```

2. Получение данных с датчика

```
90 humidity = bme280_data.humidity
91 pressure = bme280_data.pressure
92 temperature = bme280_data.temperature
```

3. Отправка данных на сервис народного мониторинга

```

103     try:
104         sock.connect(('narodmon.ru', 8283))
105         sock.send(bytearray("#{}\n{}\#\{}\n{}\#\{}\n{}\#\{}\n{}\#"
106                               .format(DEVICE_MAC,
107                                       SENSOR_ID_1, str(temperature),
108                                       SENSOR_ID_2, str(pressure),
109                                       SENSOR_ID_3, str(humidity))).encode(encoding="utf-8", errors="strict"))
110
111         data=sock.recv(1024)
112         sock.close()
113         print(data)
114     except socket.error as e:
115         print('ERROR! Exception {}'.format(e))

```

Рисунок 11 - Отправка данных на сервис "Народный мониторинг"

4. Сохранение данных в базе данных

```

116 with closing(psycopg2.connect(dbname='weather_statistic', user='weatherstation', password='weatherstation', host='192.168.1.14')) as conn:
117     with conn.cursor() as cursor:
118         cursor.execute('insert into temperature (value) values (%s)', temperature)
119         cursor.execute('insert into pressure (value) values (%s)', pressure)
120         cursor.execute('insert into humidity (value) values (%s)', humidity)

```

Рисунок 12 – Сохранение данных в БД

После выполнения этих действий скрипт прекращает свою работу.

Для того, чтобы сохранять данные в базу данных, необходимо иметь включенный и настроенный сервер PostgreSQL 11, а также доступ к нему.

Когда создан пользователь и ему предоставлены необходимые права, можно создать таблицы необходимые для работы скрипта.

Для хранения данных с датчика ВМЕ280, необходимо 3 таблицы:

- Для хранения значений температуры;

- Для хранения значений давления;
- Для хранения значений влажности.

Таблицы можно создать, используя следующие SQL скрипты:

1. Для таблицы хранения значений температуры

```

5 CREATE TABLE public.temperature
6 (
7     id bigint NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
8     value double precision,
9     timestamp timestamp without time zone NOT NULL DEFAULT now(),
10    CONSTRAINT temperature_pk PRIMARY KEY (id)
11 )
12 WITH (
13     OIDS = FALSE
14 )
15 TABLESPACE pg_default;
16
17 ALTER TABLE public.temperature
18     OWNER to weatherstation;
19 -- Index: temperature_id_idx
20
21 -- DROP INDEX public.temperature_id_idx;
22
23 CREATE INDEX temperature_id_idx
24     ON public.temperature USING btree
25     (id ASC NULLS LAST, timestamp ASC NULLS LAST)
26     TABLESPACE pg_default;

```

Рисунок 13 - Скрипт создания таблицы хранения температуры

2. Для таблицы хранения значений давления

```

5 CREATE TABLE public.pressure
6 (
7     id bigint NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
8     value double precision,
9     timestamp timestamp without time zone NOT NULL DEFAULT now(),
10    CONSTRAINT pressure_pk PRIMARY KEY (id)
11 )
12 WITH (
13     OIDS = FALSE
14 )
15 TABLESPACE pg_default;
16
17 ALTER TABLE public.pressure
18     OWNER to weatherstation;
19 -- Index: pressure_id_idx
20
21 -- DROP INDEX public.pressure_id_idx;
22
23 CREATE INDEX pressure_id_idx
24     ON public.pressure USING btree
25     (id ASC NULLS LAST, timestamp ASC NULLS LAST)
26     TABLESPACE pg_default;

```

Рисунок 14 - Скрипт создания таблицы хранения давления

3. Для таблицы хранения значений влажности

```
5 CREATE TABLE public.humidity
6 (
7     id bigint NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
8     value double precision,
9     timestamp timestamp without time zone NOT NULL DEFAULT now(),
10    CONSTRAINT humidity_pk PRIMARY KEY (id)
11 )
12 WITH (
13     OIDS = FALSE
14 )
15 TABLESPACE pg_default;
16
17 ALTER TABLE public.humidity
18     OWNER to weatherstation;
19 -- Index: humidity_id_idx
20
21 -- DROP INDEX public.humidity_id_idx;
22
23 CREATE INDEX humidity_id_idx
24     ON public.humidity USING btree
25     (id ASC NULLS LAST, timestamp ASC NULLS LAST)
26     TABLESPACE pg_default;
```

Рисунок 15 - Скрипт создания таблицы хранения давления

4.5 Установка и настройка веб сервера с движком PHP

Для того, чтобы можно было посмотреть состояние окружающей среды, необходимо реализовать инструмент, позволяющий посмотреть показания датчиков. Этот инструмент был выполнен с использованием WEB технологий. В качестве инструментов для разработки были использованы HTML, JavaScript, CSS и PHP 7. Для визуализации данных была применена библиотека ChartJS, которая позволяет построить практически неограниченное количество типов диаграмм.

Разработанная веб-панель довольно легка в использовании и имеет простую реализацию. Для понимания принципа работы интерфейса, можно рассмотреть следующие участки кода:

- Заголовок HTML файла

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Weatherstation charts</title>
6
7     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC120hlp9ysBridgE120v2Wj+8vzsqO4560v81B47855hr976Y3A2O" crossorigin="anonymous">
8
9
10    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-JQ6iE+2NNSTdQuAT2ggUQQebG8V2sQJ4N1c87HHQD7w4XWijIR840C785M97" crossorigin="anonymous"></script>
11    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-JQ6iE+2NNSTdQuAT2ggUQQebG8V2sQJ4N1c87HHQD7w4XWijIR840C785M97" crossorigin="anonymous"></script>
12    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-JQ6iE+2NNSTdQuAT2ggUQQebG8V2sQJ4N1c87HHQD7w4XWijIR840C785M97" crossorigin="anonymous"></script>
13    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-JQ6iE+2NNSTdQuAT2ggUQQebG8V2sQJ4N1c87HHQD7w4XWijIR840C785M97" crossorigin="anonymous"></script>
14    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-JQ6iE+2NNSTdQuAT2ggUQQebG8V2sQJ4N1c87HHQD7w4XWijIR840C785M97" crossorigin="anonymous"></script>
15 </head>
```

Рисунок 16 - index.php header

Данный участок кода указывает на то, какие зависимости необходимы для корректной работы веб страницы.

- Начало самого интерфейса

```

16 <body>
17 <nav>
18 <div class="nav nav-tabs" id="nav-tabs" role="tablist">
19 <a class="nav-item nav-link active" id="nav-home-tab" data-toggle="tab" href="#nav-home" role="tab" aria-controls="nav-home" aria-selected="true">Temperature/A3
20 <a class="nav-item nav-link" id="nav-profile-tab" data-toggle="tab" href="#nav-profile" role="tab" aria-controls="nav-profile" aria-selected="false">Humidity/A4
21 <a class="nav-item nav-link" id="nav-contact-tab" data-toggle="tab" href="#nav-contact" role="tab" aria-controls="nav-contact" aria-selected="false">Pressure/A5
22 </div>
23 </nav>
24 <div class="tab-content" id="nav-tabContent">
25 <div class="tab-pane fade show active" id="nav-home" role="tabpanel" aria-labelledby="nav-home-tab">
26 <canvas id="temperatureChart"></canvas>
27 </div>
28 <div class="tab-pane fade" id="nav-profile" role="tabpanel" aria-labelledby="nav-profile-tab">
29 <canvas id="humidityChart"></canvas>
30 </div>
31 <div class="tab-pane fade" id="nav-contact" role="tabpanel" aria-labelledby="nav-contact-tab">
32 <canvas id="pressureChart"></canvas>
33 </div>
34 </div>
35 </body>
36 <script>
37 var data = <?php

```

Рисунок 17- index.php зависимости

На данном участке кода расположены видимые пользователю элементы интерфейса.

Готовая аппаратно-программная система наблюдения за состоянием окружающей среды выглядит следующим образом:



Рисунок 18- Внешний вид готовой системы. Raspberry Pi

На картинке выше изображена стойка с несколькими Raspberry Pi 3 две из которых задействованы как активные участники разрабатываемой системы. Так же можно увидеть наличие модема, используемого для связи с сетью интернет.

Для подключения датчика была использована розетка с двумя разъёмами Ethernet и распаяна под поставленные цели. Данная розетка подключается к колодке портов ввода-вывода на Raspberry Pi 3. Розетка подключается к порту I2C для коммуникации с датчиком, который вынесен на улицу.



Рисунок 19 - Внешний вид готовой системы. Датчик

На данном изображении можно увидеть кожух датчика, который сделан для того, чтобы оградить сам датчик от разрушающего воздействия окружающей среды, а также исключить попадание прямых солнечных лучей для большей точности измерения.

Представленный прототип вынесен за пределы жилого помещения и отведён от стен для более точного измерения показателей воздуха.

- Php скрипт выполняющий запрос данных и формирование JSON объекта

```

36 <script>
37 var data = <?php
38 $dbconn = pg_connect("host=192.168.1.14 port=5432 dbname=weather_statistic user=weatherstation password=weatherstation")
39 or var_dump('Не удалось соединиться: ' . pg_last_error());
40
41 $dates = array();
42 $temperatureValues = array();
43 $temperatureQuery = 'SELECT * FROM temperature ORDER BY ID DESC LIMIT 10';
44 $temperatureResult = pg_query($temperatureQuery) or die('Ошибка запроса: ' . pg_last_error());
45
46 while ($line = pg_fetch_array($temperatureResult, null, PGSQL_ASSOC)) {
47
48     $humidityValues = array();
49     $humidityQuery = 'SELECT * FROM humidity ORDER BY ID DESC LIMIT 10';
50     $humidityResult = pg_query($humidityQuery) or die('Ошибка запроса: ' . pg_last_error());
51
52     while ($line = pg_fetch_array($humidityResult, null, PGSQL_ASSOC)) {
53
54         $pressureValues = array();
55         $pressureQuery = 'SELECT * FROM pressure ORDER BY ID DESC LIMIT 10';
56         $pressureResult = pg_query($pressureQuery) or die('Ошибка запроса: ' . pg_last_error());
57
58         while ($line = pg_fetch_array($pressureResult, null, PGSQL_ASSOC)) {
59
60             $result = "{dates:[".implode(",", $dates)."],temperature:{values:[".implode(",", $temperatureValues)."],pressure:{values:[".implode(",", $pressureValues)."]}}}";
61             echo $result;
62
63             pg_free_result($temperatureResult);
64             pg_free_result($humidityResult);
65             pg_free_result($pressureResult);
66
67             pg_close($dbconn);
68         }
69     }
70 }
71
72 $result = "{dates:[".implode(",", $dates)."],temperature:{values:[".implode(",", $temperatureValues)."],pressure:{values:[".implode(",", $pressureValues)."]}}}";
73 echo $result;
74
75 pg_free_result($temperatureResult);
76 pg_free_result($humidityResult);
77 pg_free_result($pressureResult);
78
79 pg_close($dbconn);
80
81 -?>
82 var ctx = document.getElementById('temperatureChart').getContext('2d');
83 var temperatureChart = new Chart(ctx, {

```

Рисунок 20- index.php PHP вставка

На скриншоте выше расположен код начала JavaScript скрипта, а также PHP вставка, которая формирует массив данных необходимый для отрисовки диаграмм.

- JavaScript сценарий позволяющий инициализировать диаграммы на основе полученных данных.

```

36  <script>
37  var data = <?php
82  var ctx = document.getElementById('temperatureChart').getContext('2d');
83  var temperatureChart = new Chart(ctx, {
100  var ctx = document.getElementById('humidityChart').getContext('2d');
101  var temperatureChart = new Chart(ctx, {
118  var ctx = document.getElementById('pressureChart').getContext('2d');
119  var temperatureChart = new Chart(ctx, {
136  </script>
137  </body>
138  </html>

```

Рисунок 21 - Инициализация диаграмм

На данном рисунке можно увидеть участок JavaScript кода, который инициализирует отрисовку диаграмм на интерфейсе, используя сгенерированный объект с данными.

5. Тестирование системы

Для того, чтобы убедиться, что система работает, необходимо установить датчик вдали от предметов, которые могут повлиять на показания каким-либо образом. Для тестирования системы датчик был помещён в защитный кожух и вынесен на улицу для измерения показаний окружающей среды вне помещения.

Чтобы протестировать систему, необходимо запустить её и подождать некоторое время для того, чтобы скрипт отработал и отправил данные в БД и на сервер народного мониторинга. После первого запуска скрипта, можно увидеть, что данные были отправлены успешно:

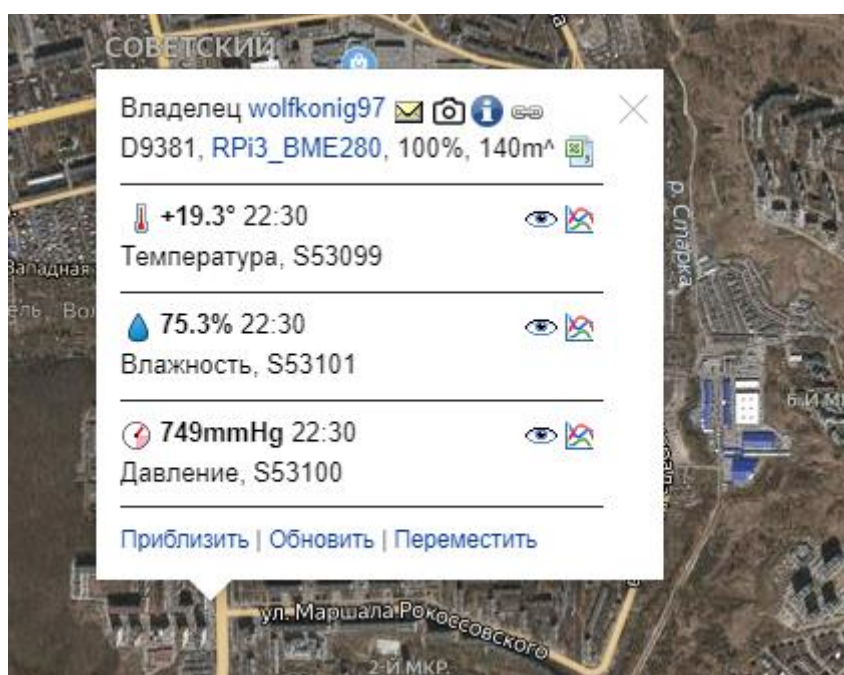


Рисунок 22 - Показания датчиков на сервисе народного мониторинга

На скриншоте выше, видно, что данные успешно были переданы на сервис народного мониторинга.

После отправки данных они так же сохраняются в базе данных, в соответствующих таблицах.

	id [PK] bigint	value double precision	timestamp timestamp without time zone
1	19484	19.2789623759745	2020-06-17 22:30:01.565976
2	19483	19.4083650276298	2020-06-17 22:25:01.864563
3	19482	19.5429439783096	2020-06-17 22:20:02.084312
4	19481	19.5015250404524	2020-06-17 22:15:02.224176

Рисунок 23 - Данные о температуре в БД

	id [PK] bigint	value double precision	timestamp timestamp without time zone
1	19484	998.722708541709	2020-06-17 22:30:01.565976
2	19483	998.860354123972	2020-06-17 22:25:01.864563
3	19482	998.954370940666	2020-06-17 22:20:02.084312
4	19481	998.068120457482	2020-06-17 22:15:02.224176

Рисунок 24 - Данные о давлении в БД

	id [PK] bigint	value double precision	timestamp timestamp without time zone
1	19484	75.2967944587935	2020-06-17 22:30:01.565976
2	19483	74.3392125792706	2020-06-17 22:25:01.864563
3	19482	73.6540193913378	2020-06-17 22:20:02.084312
4	19481	73.4040076104458	2020-06-17 22:15:02.224176

Рисунок 25 - Данные о влажности в БД

Исходя из скриншотов выше, можно видеть, то, что данные были успешно записаны на сервер баз данных. Так же можно видеть, что в базе данных значения имеют больше знаков после запятой, чем на сервисе народного мониторинга.

В том числе, необходимо убедиться в том, что веб интерфейс так же функционирует корректно. Для этого необходимо перейти по сетевому адресу, который принадлежит Raspberry Pi.

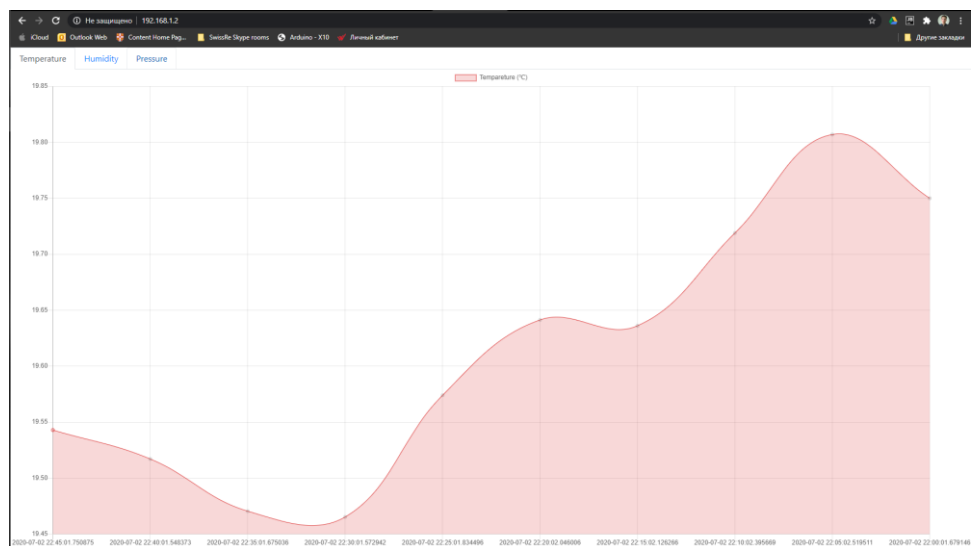


Рисунок 26 – Веб-интерфейс. График температуры.

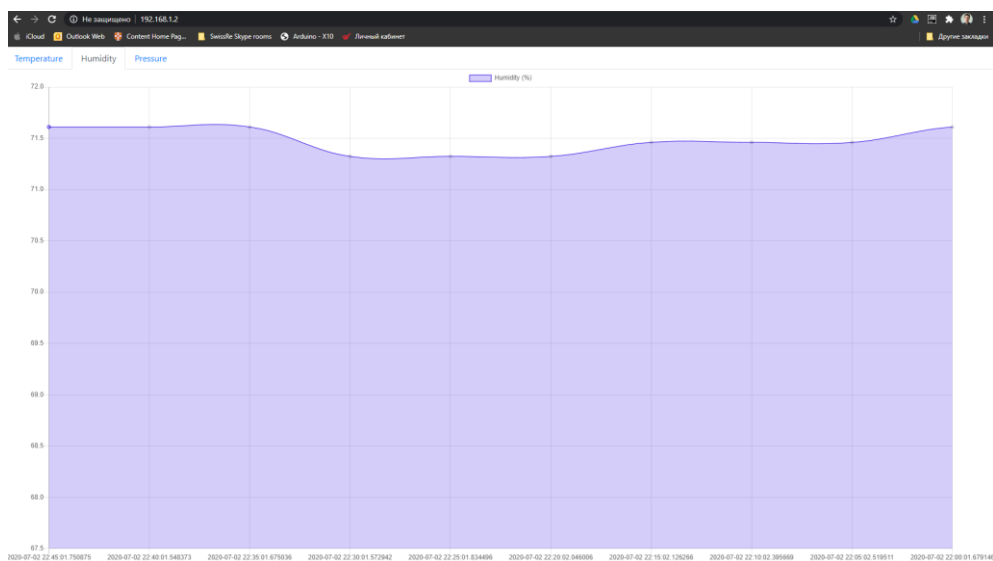


Рисунок 277 – Веб-интерфейс. График влажности.

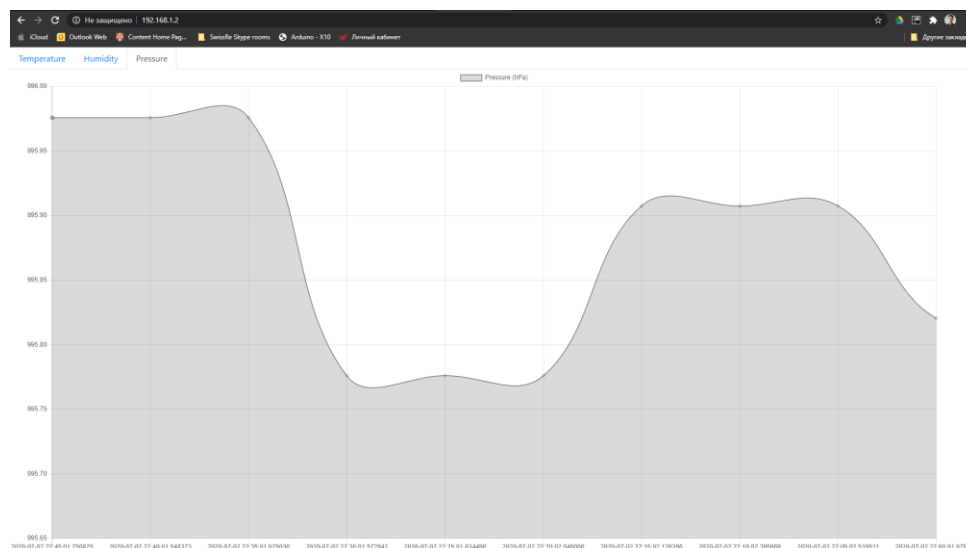


Рисунок 28 – Веб-интерфейс. График давления.

На представленных картинках выше можно увидеть демонстрационный веб-интерфейс, который визуализирует последние десять замеров. Так как замеры происходят каждые 5 минут, то это значит, что на графике расположены замеры за последние 10 минут.

После проверки работоспособности системы можно сравнить показания со значениями из сторонних источников. Например, от источника Gismeteo.

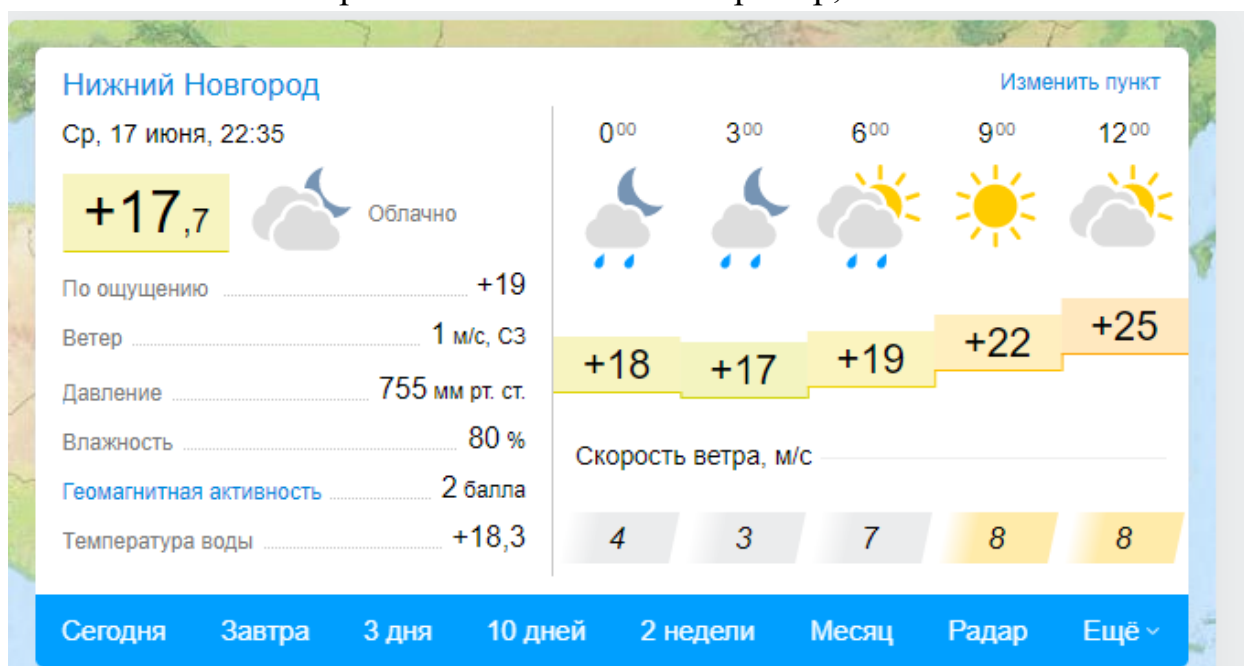


Рисунок 299 - Данные о состоянии окружающей среде от стороннего источника

На скриншоте выше, видно, что параметры, предоставленные датчиком ВМЕ280 и сторонним сервисом, различаются. Это может быть вызвано как различием местности, на которой происходит замер, так и погрешностью датчиков, используемых при замерах. Для чистоты эксперимента необходимо иметь ртутный градусник, чтобы проверить корректность данных, но таковой, к сожалению, отсутствует.

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						34
Изм	Лист	№ докум.	Подп.	Дата		

Заключение

В процессе выполнения выпускной квалификационной работы была разработана аппаратно-программная система наблюдения за изменениями окружающей среды, позволяющая наблюдать за некоторыми характеристиками окружающей в реальном времени. В том числе был реализован функционал, позволяющий хранить исторические данные и делиться ими со сторонним сервисом.

Данная система может быть полезна как для небольших домашних автоматизаций, так и послужить макетом для IoT решений.

В будущем можно добавить к системе другие типы датчиков такие как датчик направления и силы ветра, датчик уровня осадков, датчик уровня радиации, датчик шума, освещения и так далее.

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						35
Изм	Лист	№ докум.	Подп.	Дата		

Список литературы

1. Саммерфильд Марк «Программирование на Python 3. Подробное руководство» - Символ 2019
2. Доусон Майкл «Программируем на Python» - Питер 2014
3. Скляр Дэвид «Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов» - Вильямс 2017
4. OpenNet [Электронный ресурс]. [https://www.opennet.ru/ ссылка](https://www.opennet.ru/)
5. Python 3 для начинающих [Электронный ресурс]. <https://pythonworld.ru/>
6. Python Developer's Guide [Электронный ресурс]. <https://devguide.python.org/>

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						36
Изм	Лист	№ докум.	Подп.	Дата		

Приложение 1

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Weatherstatiob charts</title>

<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYYxFfc+NcPb1dK
Gj7Sk" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXa
Rkfj" crossorigin="anonymous"></script>

<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0J
KI" crossorigin="anonymous"></script>

					БКР-НГТУ-09.03.01-(15-ВМ-е)-011-2020 (ПЗ)	Лист
						37
Изм.	Лист	№ докум.	Подп.	Дата		


```
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.min.js"
integrity="sha256-R4pqcOYV8lt7snxMQO/HSbVCFRPMdrhAFMH+vr9giYI="
crossorigin="anonymous"></script>
```

```
</head>
```

```
<body>
```

```
<nav>
```

```
<div class="nav nav-tabs" id="nav-tab" role="tablist">
```

```
<a class="nav-item nav-link active" id="nav-home-tab" data-
toggle="tab" href="#nav-home" role="tab" aria-controls="nav-home" aria-
selected="true">Temperature</a>
```

```
<a class="nav-item nav-link" id="nav-profile-tab" data-toggle="tab"
href="#nav-profile" role="tab" aria-controls="nav-profile" aria-
selected="false">Humidity</a>
```

```
<a class="nav-item nav-link" id="nav-contact-tab" data-toggle="tab"
href="#nav-contact" role="tab" aria-controls="nav-contact" aria-
selected="false">Pressure</a>
```

```
</div>
```

```
</nav>
```

```
<div class="tab-content" id="nav-tabContent">
```

```
<div class="tab-pane fade show active" id="nav-home" role="tabpanel"
aria-labelledby="nav-home-tab">
```

```
<canvas id="temperatureChart"></canvas>
```

```
</div>
```

```
<div class="tab-pane fade" id="nav-profile" role="tabpanel" aria-
labelledby="nav-profile-tab">
```

```
<canvas id="humidityChart"></canvas>
```

```
</div>
```

					БКР-НГТУ-09.03.01-(15-ВМ-е)-011-2020 (ПЗ)	Лист
						38
Изм	Лист	№ докум.	Подп.	Дата		

```
<div class="tab-pane fade" id="nav-contact" role="tabpanel" aria-  
labelledby="nav-contact-tab">
```

```
<canvas id="pressureChart"></canvas>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
var data = <?php
```

```
$dbconn = pg_connect("host=192.168.1.14 port=5432 dbname=weather_statistic  
user=weatherstation password=weatherstation")
```

```
or var_dump('Не удалось соединиться: ' . pg_last_error());
```

```
$dates = array();
```

```
$temperatureValues = array();
```

```
$temperatureQuery = 'SELECT * FROM temperature ORDER BY ID DESC LIMIT  
10';
```

```
$temperatureResult = pg_query($temperatureQuery) or die('Ошибка запроса: ' .  
pg_last_error());
```

```
while ($line = pg_fetch_array($temperatureResult, null, PGSQL_ASSOC)) {
```

```
    array_push( $dates, "".$line['timestamp']. "");
```

```
    array_push( $temperatureValues, $line['value']);
```

```
}
```

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						39
Изм	Лист	№ докум.	Подп.	Дата		

```
$humidityValues = array();
```

```
$humidityQuery = 'SELECT * FROM humidity ORDER BY ID DESC LIMIT 10';
```

```
$humidityResult = pg_query($humidityQuery) or die('Ошибка запроса: ' .  
pg_last_error());
```

```
while ($line = pg_fetch_array($humidityResult, null, PGSQL_ASSOC)) {
```

```
    foreach ($line as $col_value) {
```

```
        //          var_dump($line);
```

```
        array_push( $humidityValues, $line['value']);
```

```
    }
```

```
}
```

```
$pressureValues = array();
```

```
$pressureQuery = 'SELECT * FROM pressure ORDER BY ID DESC LIMIT 10';
```

```
$pressureResult = pg_query($pressureQuery) or die('Ошибка запроса: ' .  
pg_last_error());
```

```
while ($line = pg_fetch_array($pressureResult, null, PGSQL_ASSOC)) {
```

```
    foreach ($line as $col_value) {
```

```
        array_push( $pressureValues, $line['value']);
```

```
    }
```

					БКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						40
Изм	Лист	№ докум.	Подп.	Дата		

```
}
```

```
$result = "{dates:[".implode(",", $dates)."],temperature:{values:[".implode(",",  
$temperatureValues)."]},pressure:{values:[".implode(",",  
$pressureValues)."]},humidity:{values:[".implode(",", $humidityValues)."]} }";
```

```
echo $result;
```

```
//echo "".$result;
```

```
// Очистка результата
```

```
pg_free_result($temperatureResult);
```

```
pg_free_result($humidityResult);
```

```
pg_free_result($pressureResult);
```

```
// Закрытие соединения
```

```
pg_close($dbconn);
```

```
?>;
```

```
var ctx = document.getElementById('temperatureChart').getContext('2d');
```

```
var temperatureChart = new Chart(ctx, {
```

```
    type: 'line',
```

```
    data: {
```

```
        labels: data.dates,
```

```
        datasets: [{
```

```
            label: 'Tempareture (°C)',
```

```
            data: data.temperature.values,
```

					БКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						41
Изм	Лист	№ докум.	Подп.	Дата		

```

        backgroundColor: [
            'rgba(222, 60, 60, 0.2)'
        ],
        borderColor: [
            'rgba(222, 60, 60, 1)'
        ],
        borderWidth: 1
    }]
}

});

var ctx = document.getElementById('humidityChart').getContext('2d');

var temperatureChart = new Chart(ctx, {
    type: 'line',
    data: {
        labels: data.dates,
        datasets: [{
            label: 'Humidity (%)',
            data: data.humidity.values,
            backgroundColor: [
                'rgba(41, 0, 227, 0.2)'
            ],
            borderColor: [
                'rgba(41, 0, 227, 1)'
            ],

```

					БКР-НГТУ-09.03.01-(15-ВМ-е)-011-2020 (ПЗ)	Лист
						42
Изм	Лист	№ докум.	Подп.	Дата		

```

        borderWidth: 1

    }]
}

});

var ctx = document.getElementById('pressureChart').getContext('2d');

var temperatureChart = new Chart(ctx, {

    type: 'line',

    data: {

        labels: data.dates,

        datasets: [{

            label: 'Pressure (hPa)',

            data: data.pressure.values,

            backgroundColor: [

                'rgba(68, 68, 68, 0.2)'

            ],

            borderColor: [

                'rgba(68, 68, 68, 1)'

            ],

            borderWidth: 1

        }]

    }

});

</script>

</body>

```

					БКР-ИГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						43
Изм	Лист	№ докум.	Подп.	Дата		

</html>

					БКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						44
Изм	Лист	№ докум.	Подп.	Дата		

Приложение 2

```
import bme280

import smbus2

import time

import socket

import os

import fnmatch

import datetime

import psycopg2

from contextlib import closing
```

```
port = 1
```

```
bus = smbus2.SMBus(port)
```

```
bme280_address = 0x76
```

```
I2C_ADDR = 0x27
```

```
# bme280_data = { }
```

```
DEVICE_MAC = 'B8:27:EB:A9:3A:C5'
```

```
SENSOR_ID_1 = 'T1'
```

```
SENSOR_ID_2 = 'P1'
```

```
SENSOR_ID_3 = 'H1'
```

```
# Define some device parameters
```

					БКР-ИГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						45
Изм.	Лист	№ докум.	Подп.	Дата		

LCD_WIDTH = 20 # Maximum characters per line - максимальное количество знаков в строке

Define some device constants - определим некоторые константы

LCD_CHR = 1 # Mode - Sending data - отправка данных

LCD_CMD = 0 # Mode - Sending command - отправка команды

LCD_BACKLIGHT = 0x08 # On - подсветка включена

#LCD_BACKLIGHT = 0x00 # Off - подсветка выключена

ENABLE = 0b00000100 # Enable bit

Timing constants - временные константы

E_PULSE = 0.0005

E_DELAY = 0.0005

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line - адрес в RAM дисплея для первой строки

LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line - адрес в RAM дисплея для второй строки

LCD_LINE_3 = 0x94 # LCD RAM address for the 3rd line - адрес в RAM дисплея для третьей строки

					ВКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						46
Изм.	Лист	№ докум.	Подп.	Дата		

LCD_LINE_4 = 0xD4 # LCD RAM address for the 4th line - адрес в RAM дисплея для четвёртой строки

```
def lcd_init():
```

```
    lcd_byte(0x33,LCD_CMD)
```

```
    lcd_byte(0x32,LCD_CMD)
```

```
    lcd_byte(0x06,LCD_CMD)
```

```
    lcd_byte(0x0C,LCD_CMD)
```

```
    lcd_byte(0x28,LCD_CMD)
```

```
    lcd_byte(0x01,LCD_CMD)
```

```
    time.sleep(E_DELAY)
```

```
def lcd_byte(bits, mode):
```

```
    bits_high = mode | (bits & 0xF0) | LCD_BACKLIGHT
```

```
    bits_low = mode | ((bits<<4) & 0xF0) | LCD_BACKLIGHT
```

```
    bus.write_byte(I2C_ADDR, bits_high)
```

```
    lcd_toggle_enable(bits_high)
```

```
    bus.write_byte(I2C_ADDR, bits_low)
```

```
    lcd_toggle_enable(bits_low)
```

```
def lcd_toggle_enable(bits):
```

					БКР-НГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
Изм	Лист	№ докум.	Подп.	Дата		47

```

time.sleep(E_DELAY)

bus.write_byte(I2C_ADDR, (bits | ENABLE))

time.sleep(E_PULSE)

bus.write_byte(I2C_ADDR, (bits & ~ENABLE))

time.sleep(E_DELAY)


def lcd_string(message,line):


    message = message.ljust(LCD_WIDTH," ")


    lcd_byte(line,LCD_CMD)


    for i in range(LCD_WIDTH):

        lcd_byte(ord(message[i]),LCD_CHR)


def main():

#    lcd_init()

    bme280.load_calibration_params( bus, bme280_address )

    bme280_data= bme280.sample( bus, bme280_address )


    humidity = bme280_data.humidity

    pressure = bme280_data.pressure

    temperature = bme280_data.temperature

```

					БКР-ИГТУ-09.03.01-(15-ВМ-е)-011-2020 (ПЗ)	Лист
						48
Изм	Лист	№ докум.	Подп.	Дата		

```

now = datetime.datetime.now()

#    lcd_string( "Date: " + now.strftime("%M-%d %H:%M"),LCD_LINE_1)

#    lcd_string( "Temperature: " + str("{:4.2f}".format(temperature)) + "C",
LCD_LINE_2)

#    lcd_string( "Humidity: " + str("{:4.2f}".format(humidity)) + "%",
LCD_LINE_3)

#    lcd_string( "Pressure: " + str("{:4.2f}".format(pressure)) + "hPa",
LCD_LINE_4)


sock = socket.socket()

try:

    sock.connect(('narodmon.ru', 8283))

    sock.send(bytearray("#{ }\n#{ }#{ }\n#{ }#{ }\n#{ }#{ }\n###".format(DEVICE
_MAC,

                                SENSOR_ID_1, str(temperature),

                                SENSOR_ID_2, str(pressure),

                                SENSOR_ID_3,      str(humidity)).encode(encoding="utf-8",
errors="strict"))))

    data=sock.recv(1024)

    sock.close()

    print(data)

except socket.error as e:

```

					БКР-ИГТУ-09.03.01-(15-БМ-е)-011-2020 (ПЗ)	Лист
						49
Изм	Лист	№ докум.	Подп.	Дата		

```

print('ERROR! Exception { }.format(e))

with closing(psycopg2.connect(dbname='weather_statistic',
user='weatherstation', password='weatherstation', host='192.168.1.14')) as conn:

    with conn.cursor() as cursor:

        cursor.execute('insert into temperature (value) values (%s)', temperature)

        cursor.execute('insert into pressure (value) values (%s)', pressure)

        cursor.execute('insert into humidity (value) values (%s)', humidity)

if __name__ == '__main__':

    try:

        main()

    except KeyboardInterrupt:

        pass

```

					БКР-ИГТУ-09.03.01-(15-ВМ-в)-011-2020 (ПЗ)	Лист
						50
Изм	Лист	№ докум.	Подп.	Дата		