

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий

# Курсовая работа

Колесная платформа  
по дисциплине «Микропроцессорные системы»

Выполнили:

Ферапонтов М.В.  
Савчук А.А.  
Дорошин Д.А.  
Луцай П.П.  
Артеев Д.Д.  
Яровой В.Д.  
гр. 3530904/00104

Группа:

Проверил:

Круглов С.К.

Санкт-Петербург  
2023

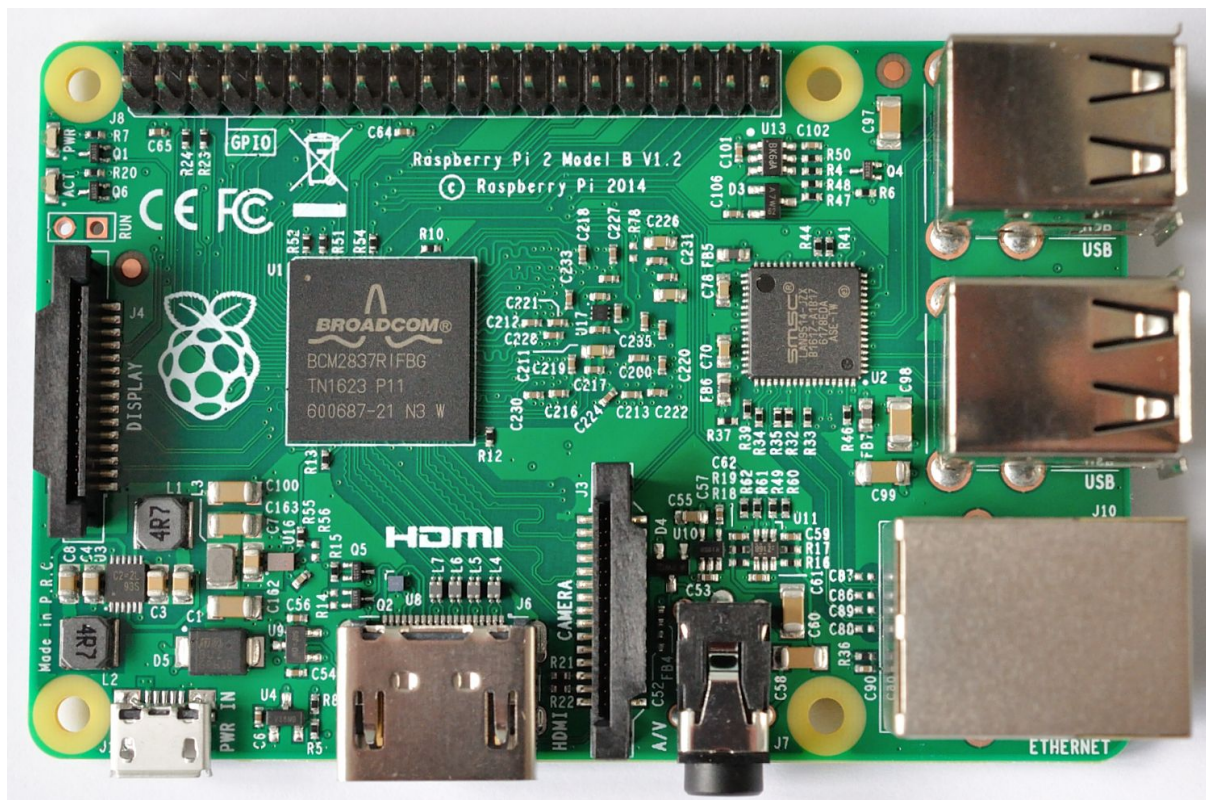
# Содержание

<b>1</b>	<b>Аппаратная реализация</b>	<b>2</b>
1.1	Raspberry pi 2b+ . . . . .	2
1.2	Используемые детали и модули . . . . .	2
1.3	Широтно-импульсная модуляция . . . . .	3
1.4	Питание . . . . .	4
<b>2</b>	<b>Схемы</b>	<b>4</b>
2.1	Принципиальная схема драйвера L289N . . . . .	4
2.2	Принципиальная схема Raspberry Pi . . . . .	6
2.3	Схема подключений . . . . .	6
<b>3</b>	<b>Rust</b>	<b>8</b>
<b>4</b>	<b>Реализация на языке Python</b>	<b>9</b>
<b>5</b>	<b>Программное обеспечение</b>	<b>10</b>
5.1	Используемые инструменты . . . . .	10
5.2	Код программы на Rust . . . . .	10

# 1 Аппаратная реализация1

## 1.1 Raspberry pi 2b+

При разработке был использован Raspberry Pi 2 model B+.



Характеристики:

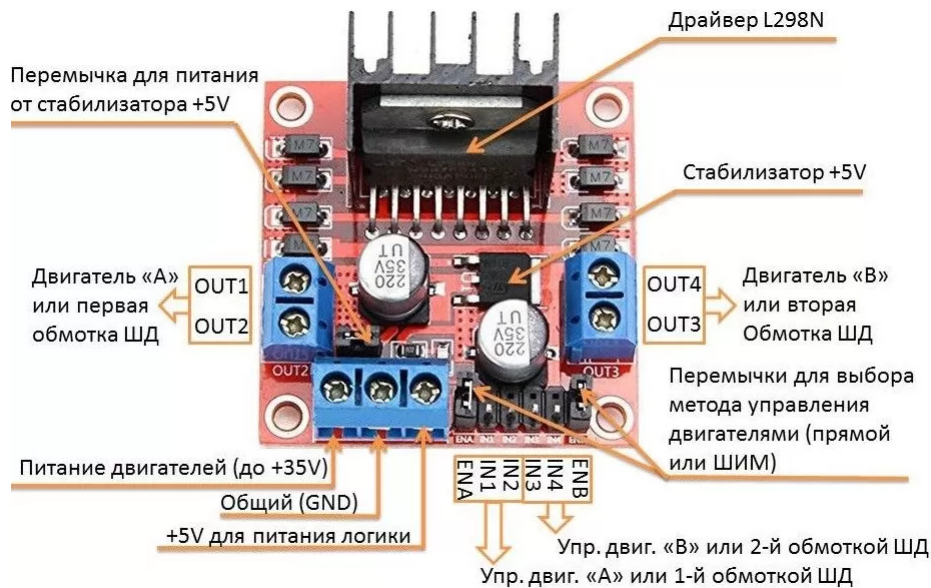
- Процессор - ARM Cortex-A7 CPU 900MHz
- Количество ядер - 4
- Оперативная память - 1Gb

## 1.2 Используемые детали и модули

Были использованы детали из набора Arduino

1. Драйвер управления движения моторов L289N
2. Колеса 4 шт.
3. Кнопка для управления движения 4шт.
4. Резистр  $22\Omega$  4 шт
5. Корпус
6. Плата расширения

Драйвер L298N используется для многофункционального управления двигателями постоянного тока. Схема модуля, состоящая из двух H-мостов, позволяет подключать к нему один биполярный шаговый двигатель или одновременно два щёточных двигателя постоянного тока. При этом есть возможность изменять скорость и направление вращения моторов. Управление осуществляется путём подачи соответствующих сигналов на командные входы, выполненные в виде штыревых контактов. На рисунке показан внешний вид модуля с кратким описанием всех его составляющих.



### 1.3 Широтно-импульсная модуляция

В драйвере L289N *ENA*, *ENB* – контакты для активации/деактивации первого и второго двигателей или соответствующих обмоток ШД. Подача логической единицы на эти контакты разрешает вращение двигателей, а логический ноль – запрещает. Для изменения скорости вращения щёточных моторов на эти контакты подаётся ШИМ-сигнал.

**ШИМ** - метод уменьшения средней мощности, передаваемой электрическим сигналом, путем эффективного разделения его на отдельные части. Среднее значение напряжения (и тока), подаваемого на нагрузку, регулируется быстрым включением и выключением переключателя между питанием и нагрузкой. Чем дольше переключатель включен по сравнению с периодами выключения, тем выше общая мощность, подаваемая на нагрузку.

```
1 pub fn set_frequency(&self, frequency: f64, duty_cycle: f64) -> Result<()>
```

- frequency - частота
- duty cycle - скважность, задается числом с плавающей точкой в промежутке между 0.0 и 1.0.

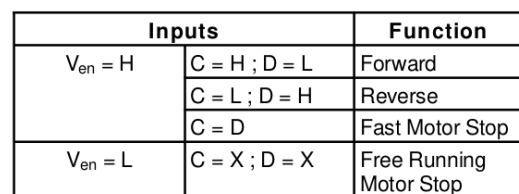
В нашей программе изначально значение скважности установлено на 0, что означает что на двигатели будет подаваться в среднем 0V. Никакого движения не происходит.

The timing diagram shows five digital signals over time, marked by vertical dashed lines. The signals are:

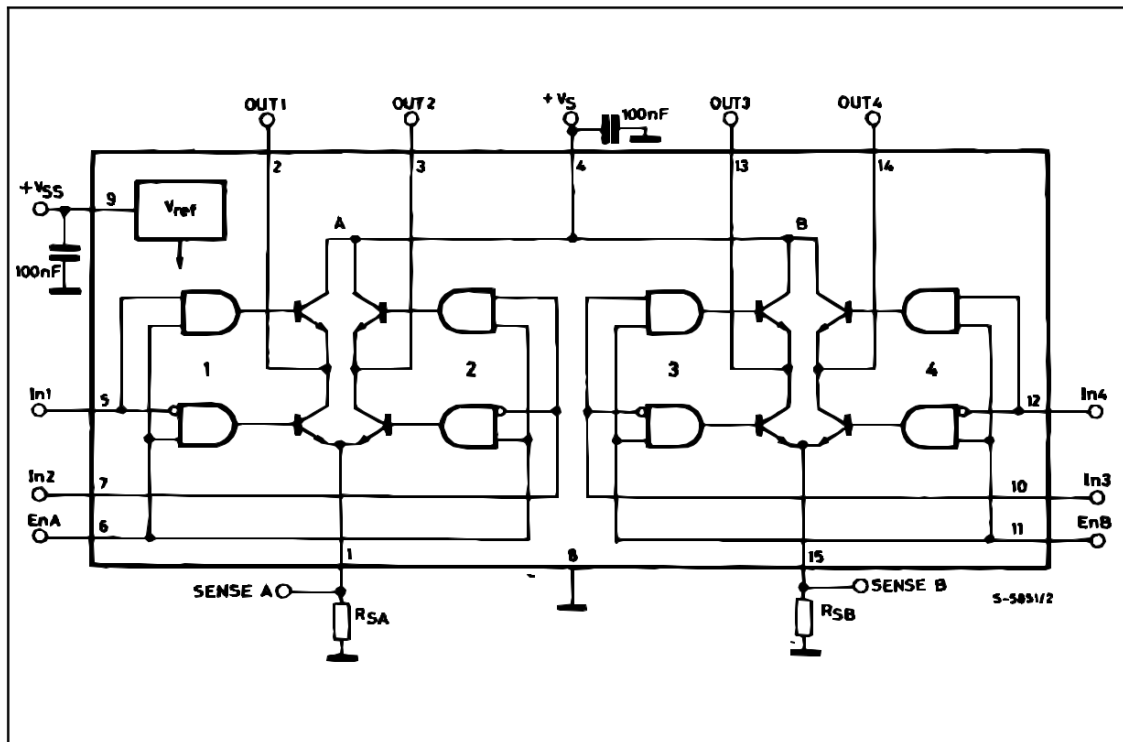
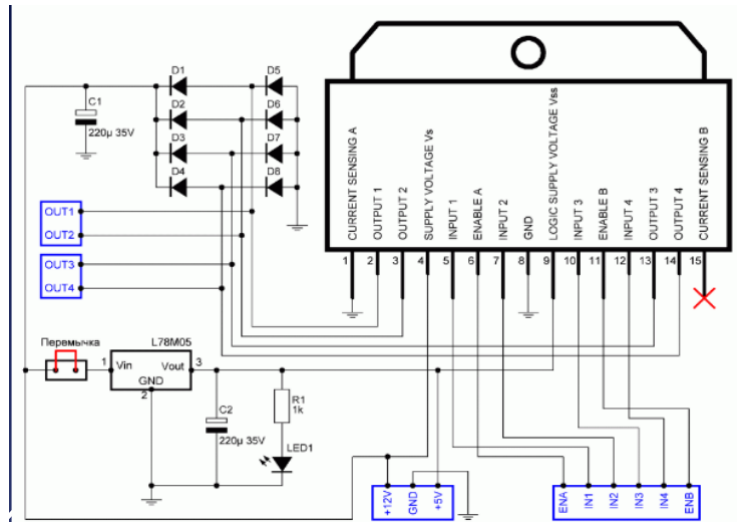
- duty cycle**: A periodic square wave that is high for approximately 1/3 of each period.
- in1**: A square wave that is high during the first two-thirds of each period.
- in2**: A square wave that is high during the last two-thirds of each period.
- in3**: A square wave that is high during the middle third of each period.
- in4**: A square wave that is high during the first third of each period.

1. L289N - Логическое устройство - 5V
2. L289N - Моторы - 12V
3. Кнопка - 3.3V
4. Raspberry pi 2b+ - 5V

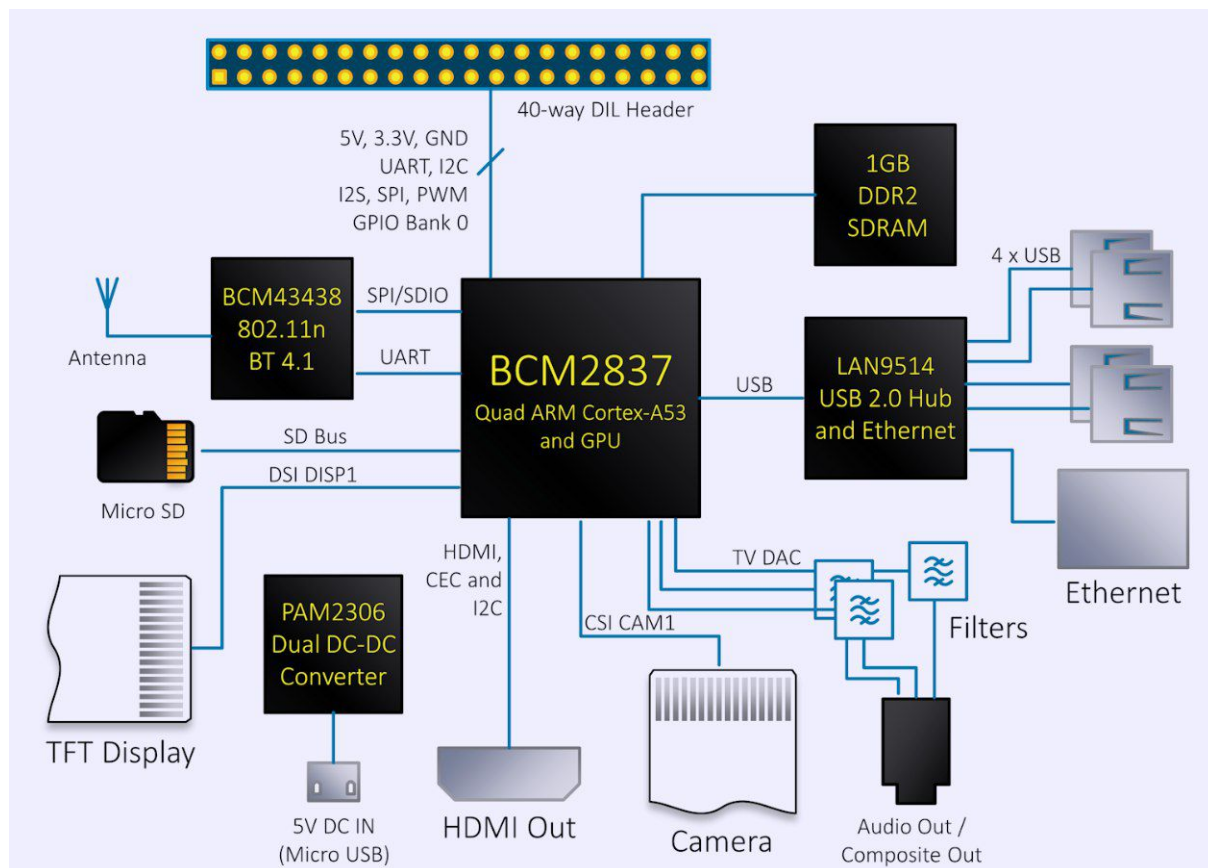
## 2.1 Принципиальная схема драйвера L289N



X = Don't care



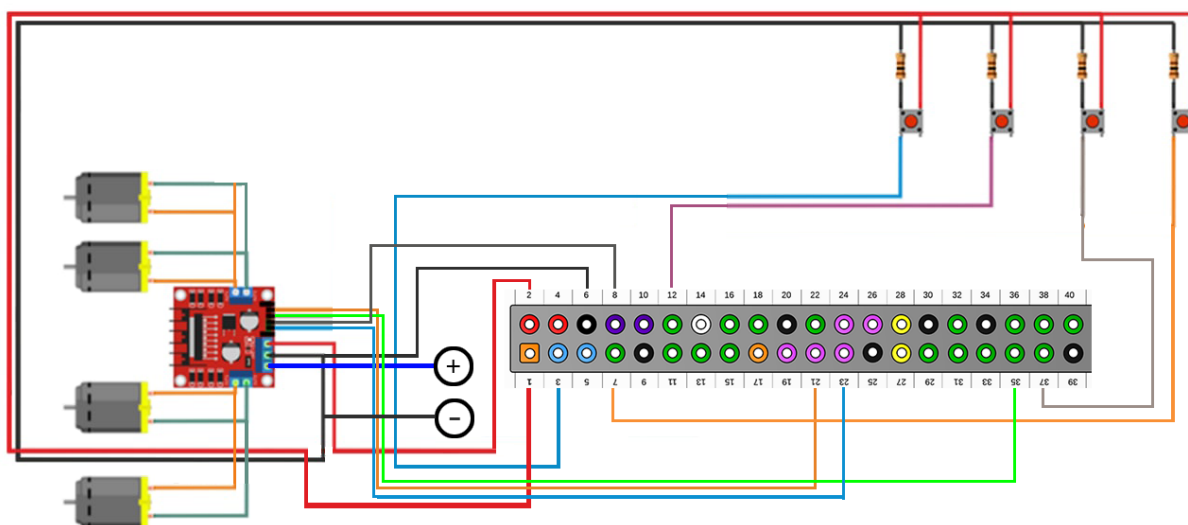
## 2.2 Принципиальная схема Raspberry Pi



Полную принципиальную схему можно найти [здесь](#).

## 2.3 Схема подключений

Пин №1 подает 3.3 V на кнопки. Пин №2 подает 5 V питания на логическое устройство. Моторы питаются из внешнего зарядного устройства.



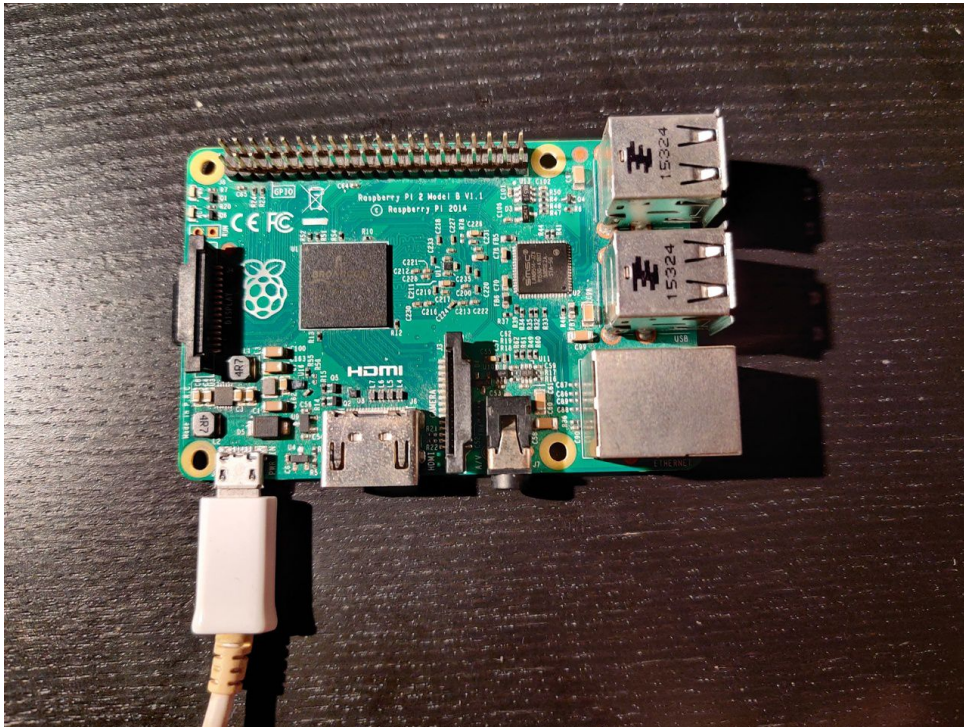


Raspberry Pi 2 Model B (J8 Header)					
GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1		2	5.0 VDC Power
8	GPIO 8 SDA1 (I2C)	3		4	5.0 VDC Power
9	GPIO 9 SCL1 (I2C)	5		6	Ground
7	GPIO 7 GPCLK0	7		8	GPIO 15 TxD (UART)
	Ground	9		10	GPIO 16 RxD (UART)
0	GPIO 0	11		12	GPIO 1 PCM_CLK/PWM0
2	GPIO 2	13		14	Ground
3	GPIO 3	15		16	GPIO 4
	3.3 VDC Power	17		18	GPIO 5
12	GPIO 12 MOSI (SPI)	19		20	Ground
13	GPIO 13 MISO (SPI)	21		22	GPIO 6
14	GPIO 14 SCLK (SPI)	23		24	GPIO 10 CE0 (SPI)
	Ground	25		26	GPIO 11 CE1 (SPI)
30	SDA0 (I2C ID EEPROM)	27		28	SCL0 (I2C ID EEPROM)
21	GPIO 21 GPCLK1	29		30	Ground
22	GPIO 22 GPCLK2	31		32	GPIO 26 PWM0
23	GPIO 23 PWM1	33		34	Ground
24	GPIO 24 PCM_FS/PWM1	35		36	GPIO 27
25	GPIO 25	37		38	GPIO 28 PCM_DIN
	Ground	39		40	GPIO 29 PCM_DOUT

**Attention!** The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

<http://www.pi4j.com>

Raspberry Pi питает от micro-USB, который подает 5V.





## 3 Rust

Для написания программы нами был выбран молодой язык программирования Rust. Но почему мы его выбрали?

**Производительность** Rust невероятно быстр и эффективно использует память: без среды выполнения или сборщика мусора он может поддерживать критически важные для производительности службы, работать на встроенных устройствах и легко интегрироваться с другими языками.

**Надежность** Богатая система типов и модель владения Rust гарантируют безопасность памяти и потокобезопасность, что позволяет устранять многие классы ошибок во время компиляции.

**Удобство** У Rust отличная документация, удобный компилятор с полезными сообщениями об ошибках и первоклассные инструменты — интегрированный менеджер пакетов и инструмент сборки, интеллектуальная поддержка нескольких редакторов с автодополнением и проверкой типов, автоматический форматировщик и многое другое.

**Первая Версия** Первый стабильный релиз, Rust 1.0, был анонсирован в мае 2015.

**Владение** Владение — это набор правил, определяющих, как программа на Rust управляет памятью. Все программы так или иначе должны использовать память компьютера во время работы. В некоторых языках есть сборщики мусора, которые регулярно отслеживают неиспользуемую память во время работы программы; в других программист должен память явно выделять и освобождать. В Rust используется третий подход: управление памятью происходит через систему владения с набором правил, которые проверяются компилятором. При нарушении любого из правил программа не будет скомпилирована.

**правило 1** У каждого значения в Rust есть владелец

**правило 2** У значения может быть только один владелец в один момент времени

**правило 3** Когда владелец покидает область видимости, значение удаляется

**Многопоточность** Безопасное и эффективное управление многопоточным программированием — ещё одна из основных целей Rust. Поэтому он обеспечивает решение с наилучшей производительностью в любой конкретной ситуации и будут иметь меньше абстракций по сравнению с аппаратным обеспечением. Поэтому Rust предлагает множество инструментов для моделирования проблем любым способом, который подходит для вашей ситуации и требований.

**способ 1** потоки для одновременного запуска нескольких фрагментов кода

**способ 2** передачи сообщений, где каналы передают сообщения между потоками

**способ 3** Многопоточность для совместно используемого состояния, когда несколько потоков имеют доступ к некоторому фрагменту данных

## 4 Реализация на языке Python

В рамках дополнительного задания нужно было реализовать программу с тем же функционалом на языке Python.

## 5 Программное обеспечение

### 5.1 Используемые инструменты

1. [Rust](#) - 1.65.0 - Язык программирования
2. [rppal](#) - 0.14.0 - Raspberry Pi Peripheral Access Library. Предоставляет доступ к GPIO у Raspberry Pi.
3. [Python](#) -3.10.1 - Язык программирования
4. [Rpi.GPIO](#) - 0.7.0 - Raspberry Pi GPIO. Предоставляет доступ к GPIO у Raspberry Pi.

### 5.2 Код программы на Rust