

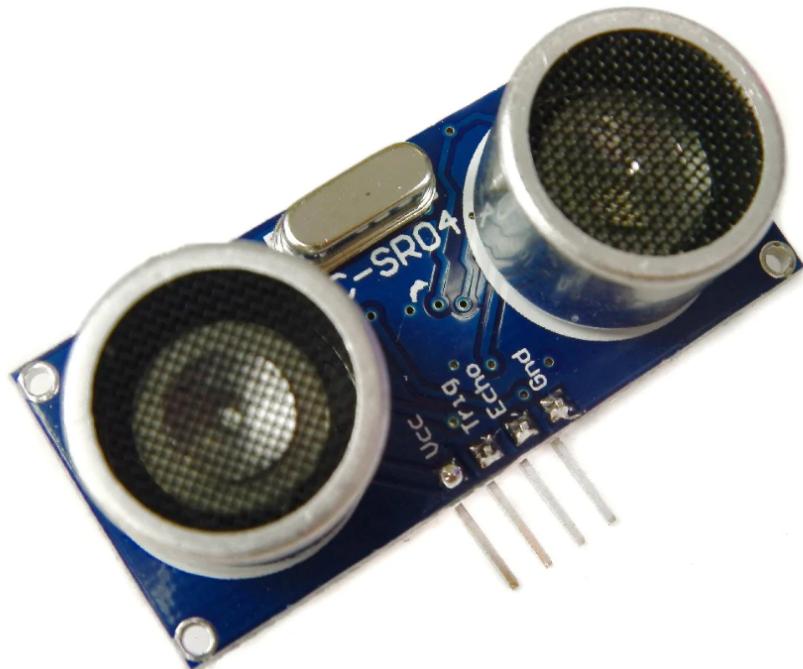
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий  
Высшая школа программной инженерии

**КУРСОВАЯ РАБОТА**  
**ОПРЕДЕЛЕНИЕ РАССТОЯНИЯ ДО ОБЪЕКТА ПУТЕМ**  
**УЛЬТРАЗВУКОВОГО ИЗМЕРЕНИЯ С ПОМОЩЬЮ HC-SR04**  
по дисциплине «Микропроцессорные системы»



Выполнили студенты группы: 3530904/00104:

Пятизянцев И. А.  
Мурзаканов И. М.  
Поздняков А. А.  
Почернин В. С.  
Шиляев В. С.

Преподаватель:

Круглов С. К.

## **Содержание**

<b>1 Условие задачи</b>	<b>2</b>
<b>2 Используемое оборудование</b>	<b>3</b>
<b>3 Ультразвуковые датчики расстояния</b>	<b>4</b>
3.1 Общее описание . . . . .	4
3.2 Описание HC-SR04 . . . . .	5
3.3 Спецификация HC-SR04 . . . . .	6
<b>4 Raspberry Pi 4B. GPIO</b>	<b>7</b>
4.1 Описание . . . . .	7
4.2 Взаимодействие . . . . .	7
<b>5 Схема устройства</b>	<b>8</b>
5.1 Делитель напряжения . . . . .	8
5.2 Схема устройства . . . . .	9
5.3 Сборка схемы . . . . .	10
<b>6 Программная часть</b>	<b>14</b>
<b>7 Raspberry Pi 4B. Удаленное управление через SSH</b>	<b>15</b>
<b>8 Результаты работы</b>	<b>16</b>
<b>9 Вывод</b>	<b>18</b>
<b>10 Исходные тексты программ</b>	<b>19</b>
10.1 main.py . . . . .	19
<b>11 Список литературы и электронных источников</b>	<b>21</b>

## **1 Условие задачи**

Поставленная задача: разработать и собрать систему на базе платы с процессором ARM, которая позволит измерить расстояние до объекта с помощью ультразвукового датчика.

## **2 Используемое оборудование**

При выборе основной платы была выбрана Raspberry Pi. В нашем случае в качестве операционной системы использовался Raspbian OS - родной дистрибутив для платы. Таким образом на данной плате получилось не только запустить проект, но и вывести результаты благодаря подключению по SSH.

Ниже представлены **технические характеристики**:

- **Процессор:** Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz;
- **Оперативная память:** 4GB LPDDR4-2400 SDRAM;
- **Цифровой видеовыход:** HDMI;
- **Композитный выход:** 3.5 мм (4 pin);
- **USB порты:** USB 2.0 x4;
- **Сеть:** WiFi 802.11n, 10/100 Мб RJ45 Ethernet;
- **Bluetooth:** Bluetooth 4.1, Bluetooth Low Energy;
- **Разъем дисплея:** Display Serial Interface (DSI);
- **Карта памяти:** MicroSD;
- **Порты ввода-вывода:** 40;
- **Модуль:** HC-SR04 Ultrasonic Range Sensor;
- **Радиодетали:**
  - $1k\Omega$  резистор;
  - $2k\Omega$  резистор;
  - Соединительные провода;

### 3 Ультразвуковые датчики расстояния

#### 3.1 Общее описание

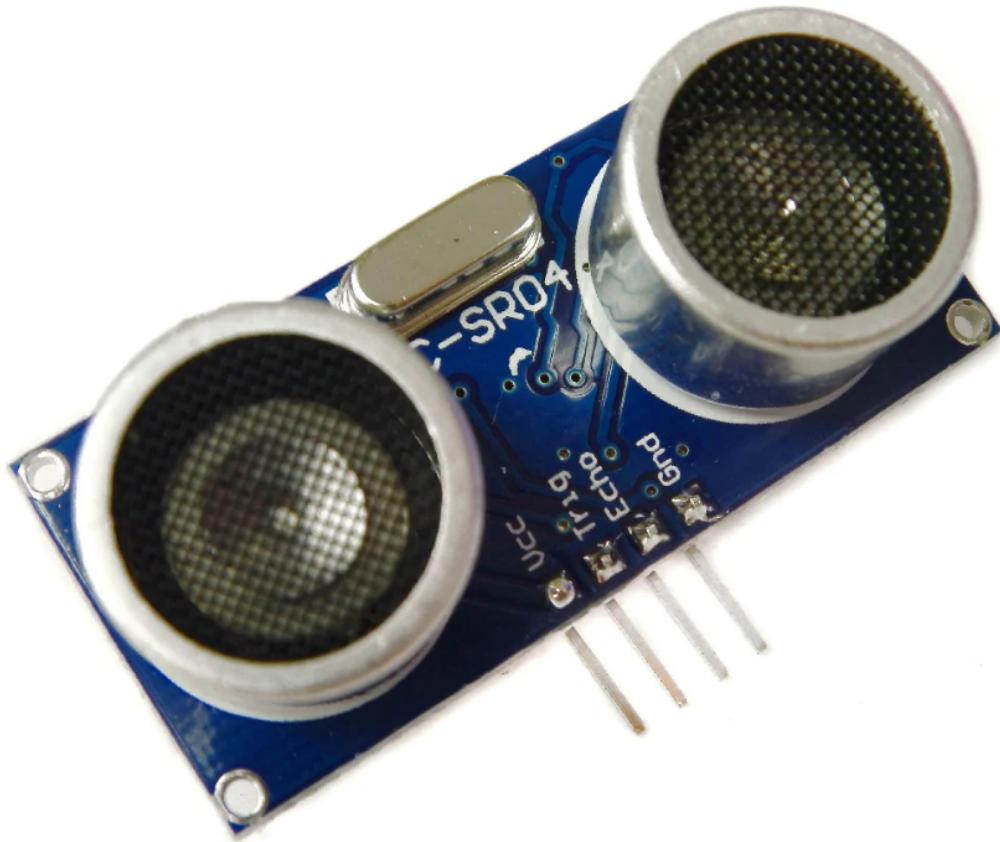


Рис. 1: Ультразвуковой датчик HC-SR04

Звук состоит из колеблющихся волн в среде (например, в воздухе), высота звука которых определяется близостью этих волн друг к другу и называется **частотой**. Человеческому уху слышна только часть звукового спектра - диапазон частот звуковых волн, определяемый как **акустический диапазон**.

Очень низкочастотный звук ниже акустического диапазона определяется как *инфразвук*, выше - как *ультразвук*. Ультразвуковые датчики предназначены для определения близости или расстояния до объекта с помощью отражения ультразвука, подобно радару. Они используют механизм расчета времени, необходимого для отражения ультразвуковых волн между датчиком и твердым объектом.

Ультразвук в основном используется, потому что он не слышим для человеческого уха и относительно точен на коротких расстояниях. Можно, конечно, использовать для этой цели акустический звук, однако, в таком случае мы получим менее точные измерения и раздражающее устройство, которое будет издавать неприятные звуки раз в секунду.

Базовый ультразвуковой датчик состоит из одного или нескольких ультразвуковых динамиков, приемника и схемы управления.

Динамики излучают высокочастотный ультразвук, который отражается от любых близлежащих твердых предметов. Часть этого ультразвукового шума от-

ражается и обнаруживается приемником на датчике. Затем этот обратный сигнал обрабатывается схемой управления для вычисления разницы во времени между передаваемым и принимаемым сигналом.

Это время можно впоследствии использовать вместе с некоторыми умными математическими вычислениями для расчета расстояния между датчиком и отражающим объектом.

### 3.2 Описание HC-SR04

Ультразвуковой датчик *HC-SR04*, который мы будем использовать для Raspberry Pi имеет четыре контакта:

- Заземление (GND);
- Выход эхо-импульса (ECHO);
- Вход триггерного импульса (TRIG);
- Источник питания 5В (VCC);

Мы питаем модуль с помощью VCC, заземляем его с помощью GND и используем наш Raspberry Pi для отправки входного сигнала на TRIG, который запускает датчик для отправки ультразвукового импульса.

Пульсовые волны отражаются от любых близлежащих объектов, а некоторые отражаются обратно к датчику. Датчик обнаруживает эти возвратные волны и измеряет время между триггером и возвратным импульсом, а затем отправляет сигнал 5В на выход ECHO.

ECHO будет «низким» (0В) до тех пор, пока датчик не сработает, когда получит TRIG импульс. Как только триггерный импульс будет обнаружен - ECHO устанавливается на «высокий уровень» (5В) на время этого импульса (т. е. пока выпущенный звуковой сигнал не вернется обратно). Длительность импульса - это полное время между отправкой датчиком ультразвукового импульса и обнаружением обратного импульса приемником датчика. Поэтому наш Python скрипт должен будет измерить длительность импульса, а затем вычислять расстояние, исходя из этого.

Однако, существует трудность. Дело в том, что выходной сигнал датчика ECHO на HC-SR04 рассчитан на 5В, в то время, как входной контакт Raspberry Pi GPIO рассчитан на 3.3В. Отправка сигнала 5В на этот незащищенный входной порт может повредить контакты GPIO.

Нам понадобится небольшая схема делителя напряжения, состоящая из двух резисторов, чтобы снизить выходное напряжение датчика до уровня, с которым может справиться Raspberry Pi.

### 3.3 Спецификация HC-SR04

Параметр	Значение
Рабочее напряжение	3,3 В ~ 5 В
Сила тока покоя	< 2 мА
Рабочая сила тока	15 мА
Рабочая частота	40 кГц
Рабочий диапазон и точность	2 см ~ 400 см ± 3 мм
Чувствительность	-65 дБ минимум
Звуковое давление	112 дБ
Эффективный угол	15°
Соединение	4-контактный разъём
Размеры	45 мм × 20 мм × 15 мм
Вес	9 г

Таблица 1: Спецификация датчика

В элементах датчика используются пьезоэлектрические кристаллы. Когда на них подаётся ток, они колеблются на высоких частотах. Пьезоэлектрические кристаллы генерируют электрический сигнал, когда ультразвуковая волна возвращается к поверхности датчика.

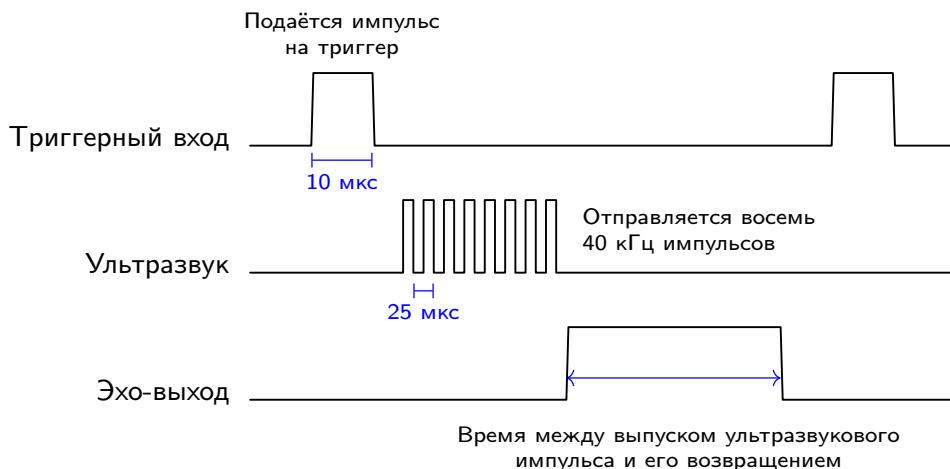


Рис. 2: Временная диаграмма работы датчика

Выше показана временная диаграмма работы датчика. Чтобы начать замеры, необходимо подать короткий импульс 10 мкс на триггерный контакт. Модуль отправит пакет из 8 ультразвуковых импульсов и переведёт эхо-выход в высокий уровень. Уровень эхо-выхода вновь станет низким, когда датчик примет отраженный звуковой сигнал.

По интервалу времени между отправкой ультразвуковых импульсов и получением эхо-сигнала можно рассчитать расстояние.

Рекомендуется использовать цикл измерения дольше 60 мс.

Для наилучших результатов поверхность объекта для обнаружения должна иметь площадь не менее  $0,5 \text{ м}^2$ .

## 4 Raspberry Pi 4B. GPIO

### 4.1 Описание

В первую очередь необходимо рассмотреть ключевые особенности этого интерфейса. И самое главное в GPIO Raspberry Pi – это pings (пины). Именно они используются для связи одноплатника с периферией. В совокупности есть 26 GPIO (портов). Все они являются цифровыми.

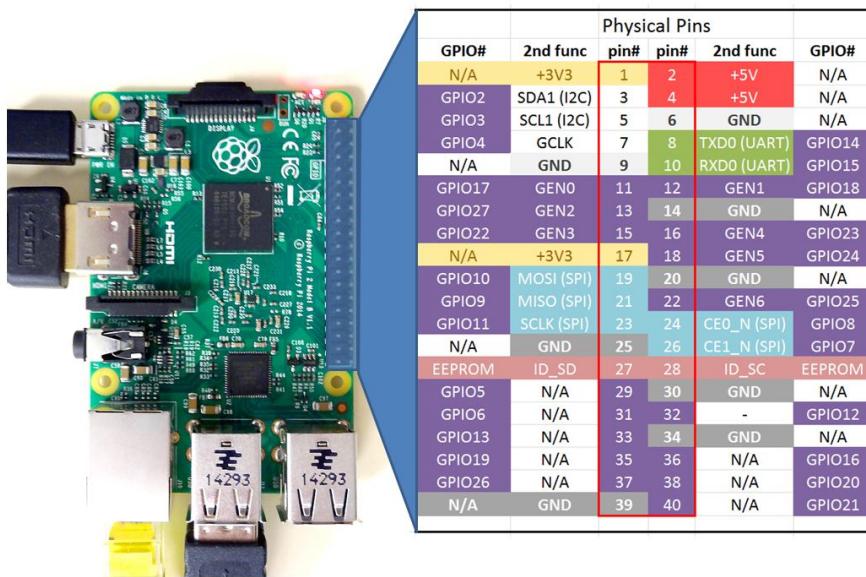


Рис. 3: Расширенная схема GPIO

### 4.2 Взаимодействие

Работать с GPIO Raspberry Pi можно практически через любой инструмент. К сегодняшнему дню созданы соответствующие библиотеки почти под все распространенные языки программирования.

Однако мы решили взаимодействовать с данным интерфейсом посредством Python. Это обусловлено, во-первых тем, что для GPIO в Raspbian уже предустановлена соответствующая библиотека для Python, а, во-вторых, этот ЯП является основным для рассматриваемого одноплатника.

Однако при желании, конечно, можно пользоваться и любыми другими инструментами. Найти название библиотек и их описание не составляет никакого труда.

## 5 Схема устройства

### 5.1 Делитель напряжения

Делитель напряжения, который мы будем использовать состоит из двух резисторов ( $R_1$  и  $R_2$ ), последовательно подключенных к входному напряжению ( $V_{in}$ ), которое необходимо уменьшить до нашего выходного напряжения ( $V_{out}$ ). В нашей схеме  $V_{in}$  будет ECHO, который нужно уменьшить на  $V_{out}$  (вход GPIO) с 5В до 3.3В.

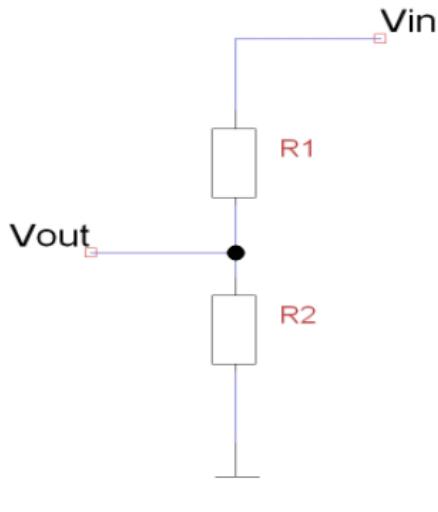


Рис. 4: Схема делителя напряжения

В действительности, нам нужно рассчитать только одно значение резистора, так как нам важен коэффициент деления:

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2}$$

Мы знаем наше входное напряжение 5В и требуемое выходное напряжение 3.3В, соответственно, можем использовать любую комбинацию резисторов для достижения снижения. Пусть  $R_1 = 1\text{k}\Omega$ , тогда получим:

$$\frac{3.3}{5} = \frac{R_2}{1000 + R_2}$$

$$0.66 = \frac{R_2}{1000 + R_2}$$

$$0.66(1000 + R_2) = R_2$$

$$660 + 0.66R_2 = R_2$$

$$660 = 0.34R_2$$

$$R_2 \approx 2\text{k}\Omega$$

## 5.2 Схема устройства

Резюмируя вышесказанное, мы можем представить следующую схему проекта:

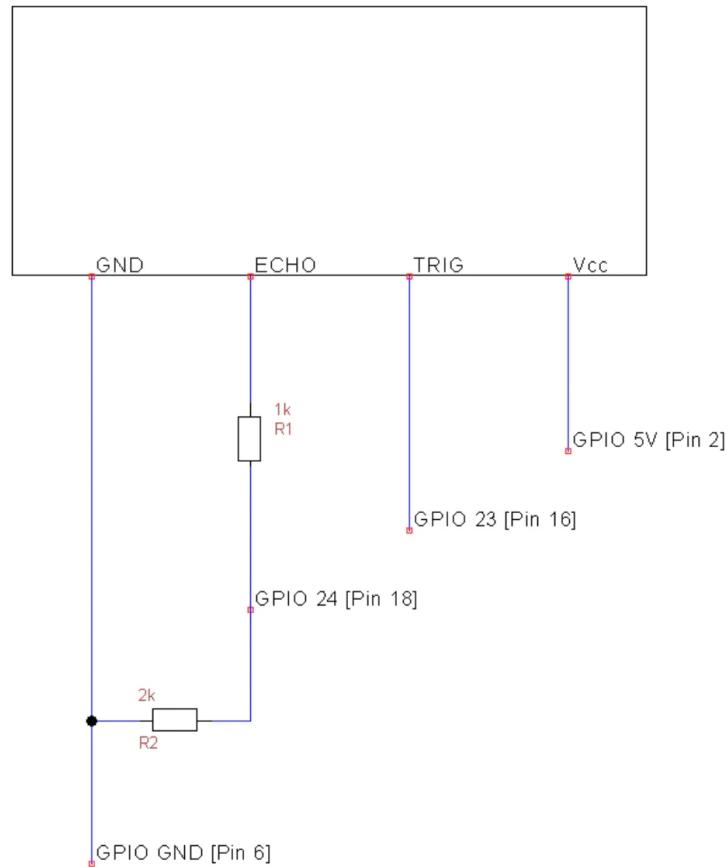


Рис. 5: Схема проекта

Еще раз покажем схему в другом виде:

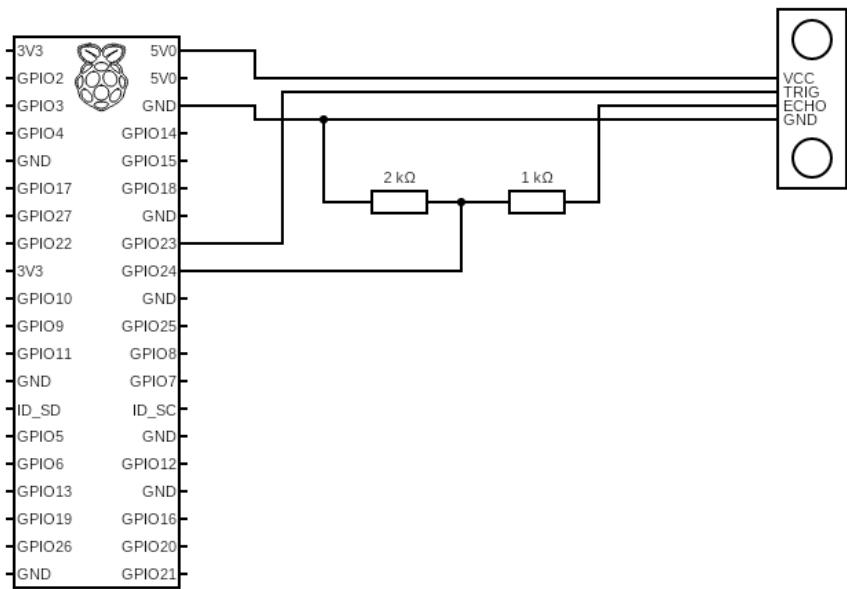


Рис. 6: Более наглядная схема

### 5.3 Сборка схемы

Рассмотрим полный алгоритм сборки схемы с использованием макетной платы:

- 1) Первым шагом подключаем перемычки типа «папа-мама» в контакты на HC-SR04 следующим образом:

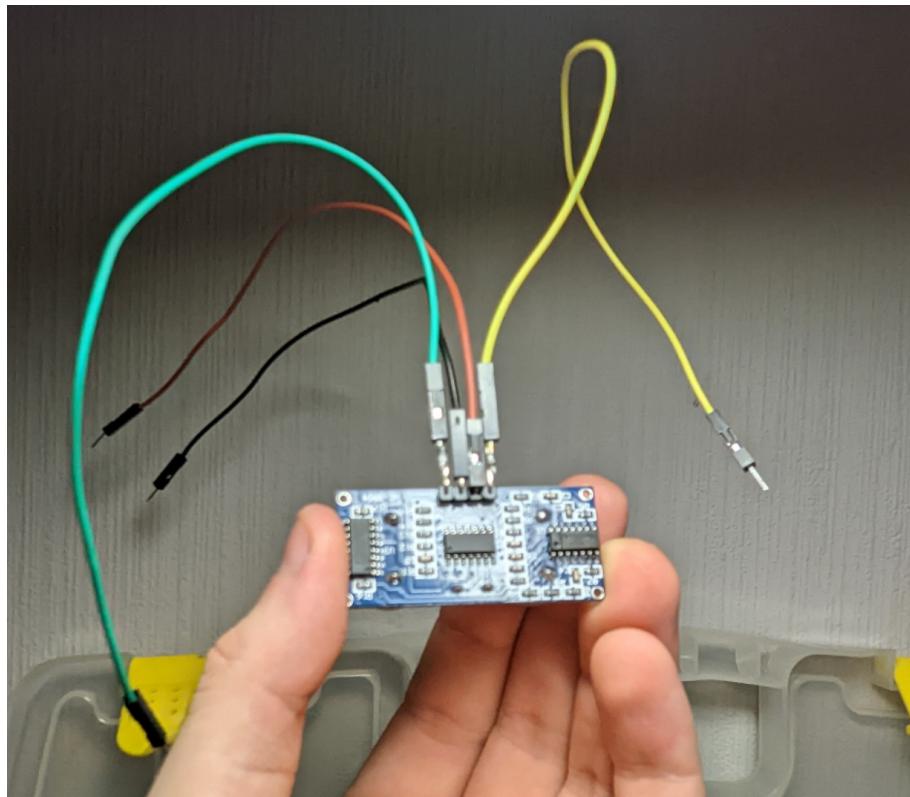


Рис. 7: Подключение перемычек к HC-SR04

- 2) Подключаем VCC к положительной шине макетной платы и GND к отрица-

тельной шине макетной платы.

- 3) Подключаем GPIO 5V к положительной шине макетной платы, а GPIO GND к отрицательной шине макетной платы.

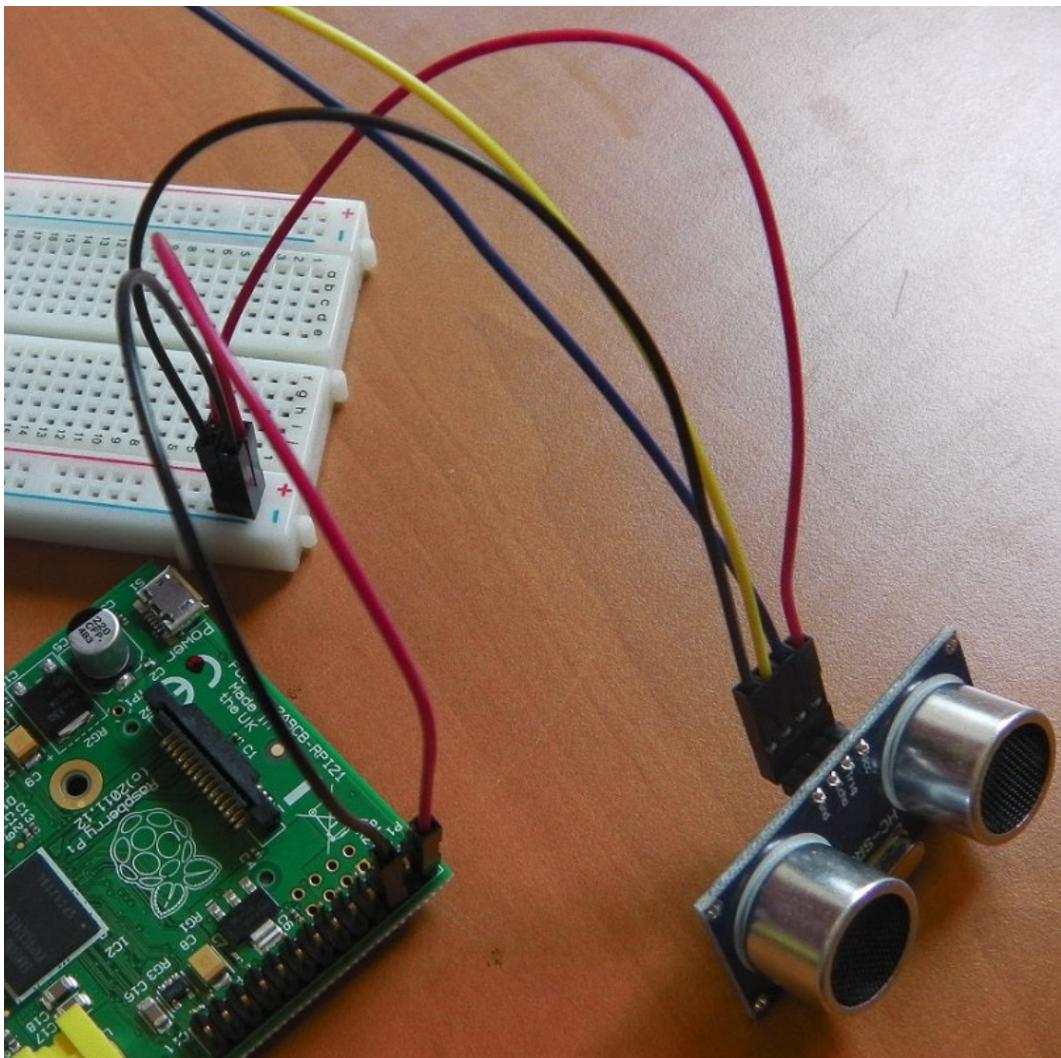


Рис. 8: Подключение контактов 5В и GND к макетной плате

- 4) Вставляем TRIG в пустую шину и подключаем эту шину к GPIO 23.

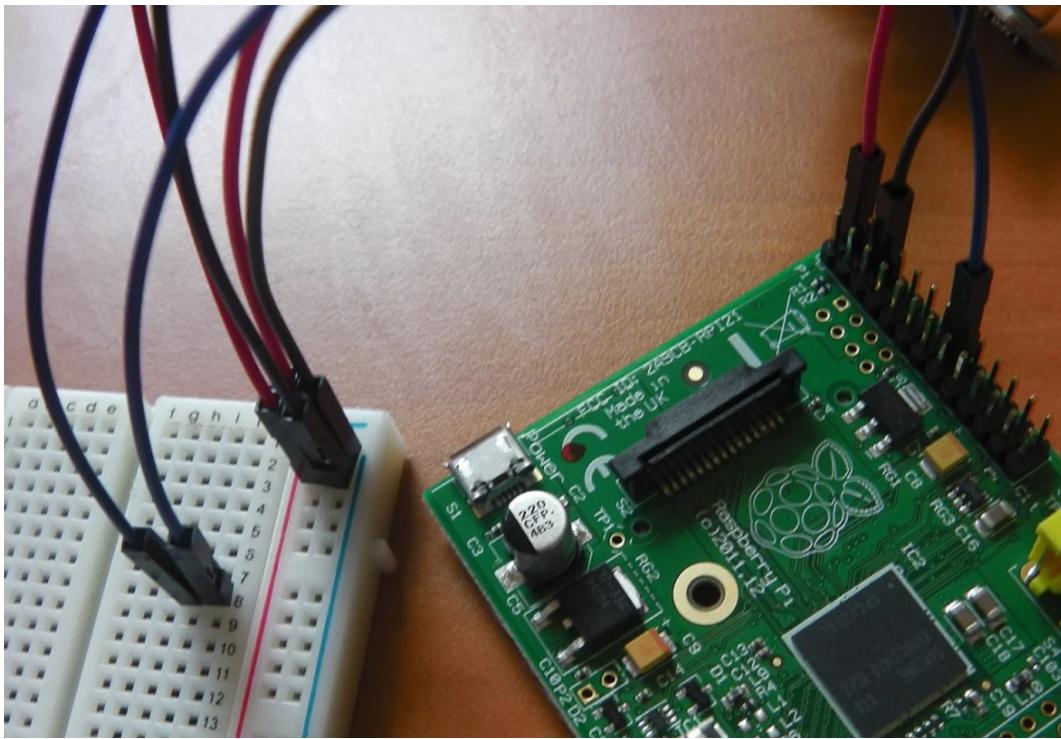


Рис. 9: Подключение TRIG

- 5) Подключаем ECHO к пустой шине, соединяя другую пустую шину с первой с помощью  $R_1$  ( $1\text{k}\Omega$ ).
- 6) Соединяя шину с  $R_1$  с шиной GND с помощью резистора  $R_2$  ( $2\text{k}\Omega$ ), оставив пространство между двумя резисторами.

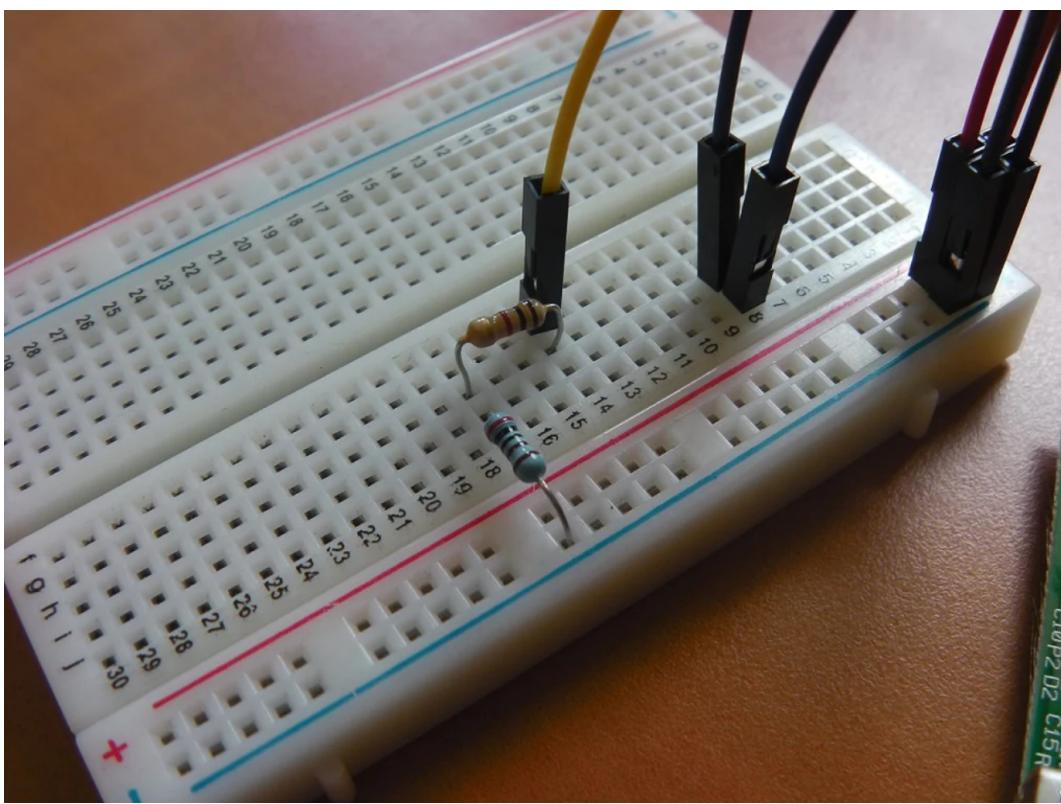


Рис. 10: Подключение резисторов к макетной плате

- 7) Подключаем GPIO 24 к шине с  $R_1$ . Этот вывод GPIO должен располагаться

междуду  $R_1$  и  $R_2$ .

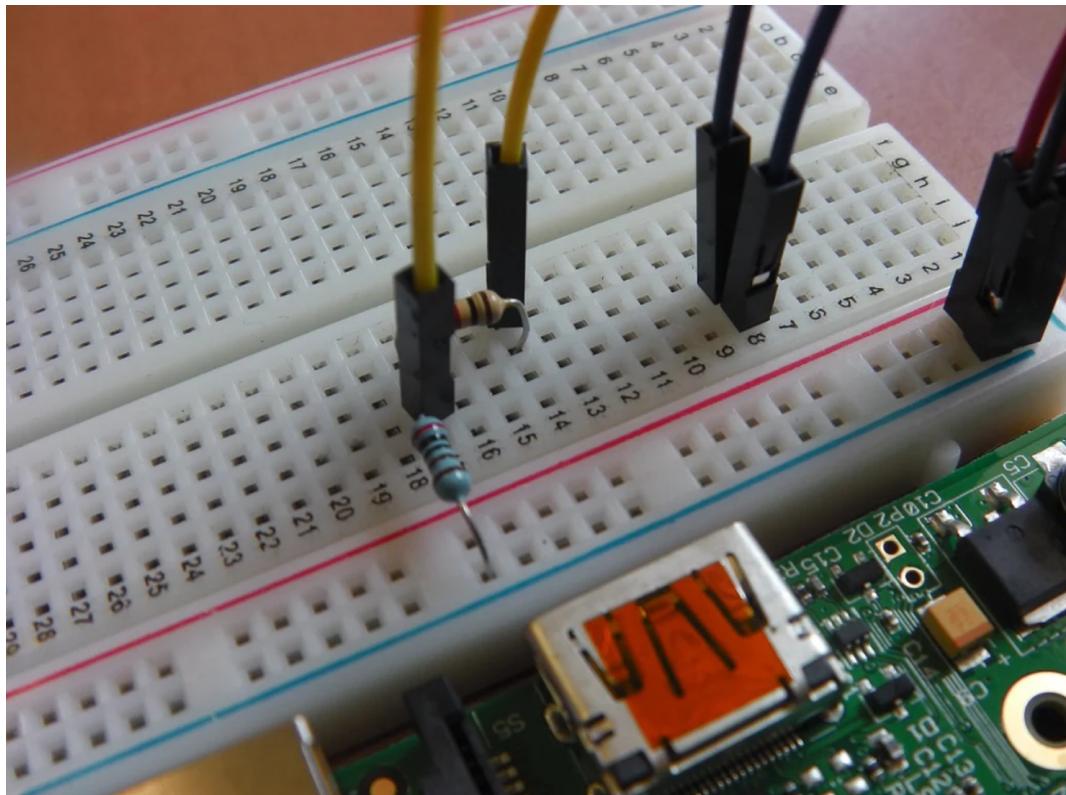


Рис. 11: Подключение GPIO 24

8) Наш датчик HC-SR04 оказывается подключенным к Raspberry Pi.

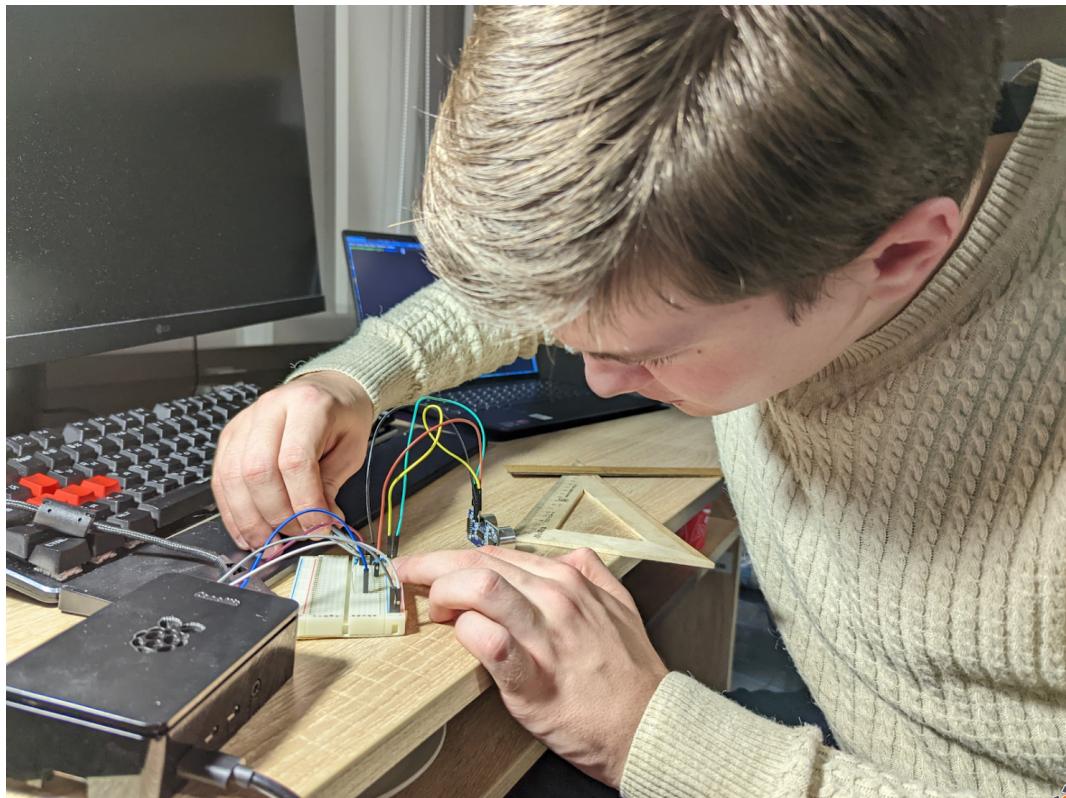


Рис. 12: Финал подключения

## 6 Программная часть

Теперь, когда мы подключили наш ультразвуковой датчик, нам нужно написать Python скрипт для определения расстояния.

Выход ультразвукового датчика ECHO будет всегда иметь низкий уровень сигнала 5В, если только он не сработал. Поэтому нам нужно установить один контакт GPIO в качестве выхода для срабатывания датчика и один в качестве входа для обнаружения изменения напряжения ECHO.

Датчику HC-SR04 требуется короткий импульс длительностью 10 мкс для запуска модуля, который заставит датчик запустить программу измерения дальности (8 ультразвуковых импульсов с частотой 40 кГц) для получения эхо-ответа. Итак, чтобы создать наш триггерный импульс, мы устанавливаем триггерный контакт в высокий уровень на 10 мкс, а затем снова устанавливаем его в низкий уровень.

Как только сигнал получен, значение ECHO изменяется с низкого (логический ноль) на высокое (логическая единица), и сигнал остается высоким на протяжении всего эхо-импульса. Поэтому нам также нужна последняя высокая метка времени для ECHO (т.е. тот момент времени, когда в последний раз сигнал был высоким).

Теперь мы можем вычислить разницу между двумя записанными метками времени и, следовательно, получить длительность импульса.

Учитывая время, необходимое сигналу для прохождения к объекту и обратно, мы можем рассчитать расстояние.

Скорость звука варьируется в зависимости от среды, через которую он проходит, а также от температуры этой среды. Однако, физики рассчитали скорость звука на уровня моря, которую мы можем взять за основу:

$$v_{\text{звук}} = 343 \frac{\text{м}}{\text{с}}$$

Нам также нужно будет разделить наше время на два, потому что сейчас мы имеем время, которое требуется ультразвуковому импульсу для прохождения расстояния до объекта и обратно.

Также, для повышения точности будем делать по три замера и выводить на экран среднее арифметическое. Выводы будем осуществлять в автоматическом режиме раз в секунду.

Исходный текст программы представлен в конце отчета.

## **7 Raspberry Pi 4B. Удаленное управление через SSH**

SSH (Secure Shell) - протокол передачи данных по сети, позволяющий производить удаленное управление операционной системой через шифрованное соединение, а также передавать файлы, транслировать видео и аудио потоки и сжимать передаваемые данные на лету. Говоря простым языком, SSH позволяет нам установить соединение, через которое можно запустить на компьютере терминал для выполнения консольных команд на другом компьютере.

Для демонстрации мы используем SSH-клиент JuiceSSH для Android.

## 8 Результаты работы

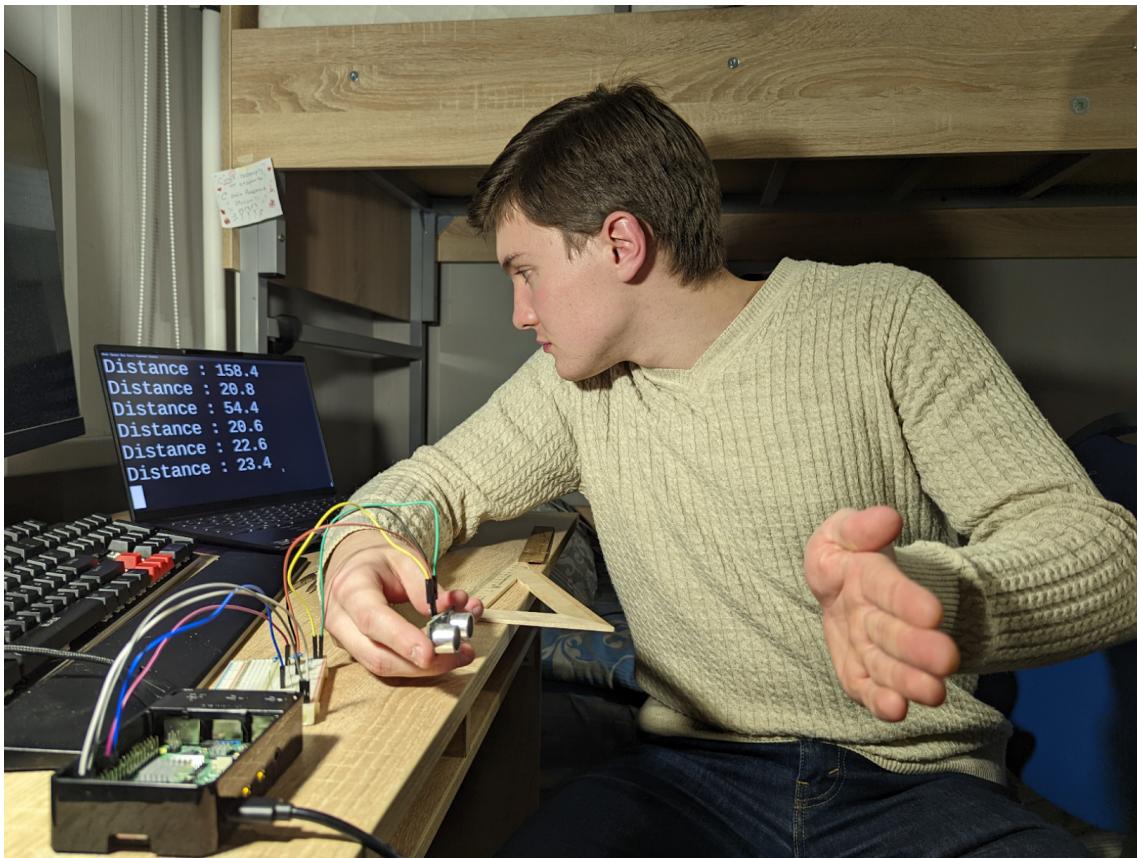


Рис. 13: Тестирование схемы

В ходе тестирования было проведено 4 измерения: 2 на малом расстоянии, 1 на среднем расстоянии и 1 на большом расстоянии. Измерения проводились в стандартном коридоре, в качестве отражающей поверхности на малом расстоянии использовалась упаковка туалетной бумаги, а в тестах на среднем и большом расстоянии - стена.

- На малом расстоянии (100мм) прибор в среднем показывал результат 103мм.
- На малом расстоянии (500мм) прибор в среднем показывал результат 503мм.
- На среднем расстоянии (1000мм) прибор вначале показывал заниженные результаты при измерении от пола, что связано с отражением ультразвука от данной поверхности. Однако при измерении с возвышенности результат вновь стал корректным, прибор показывал результат 1015мм.
- На большом расстоянии (3000мм) прибор показывал данные значительно меньше фактического расстояния (1300мм). Данный эффект объясняется тем, что в данном teste расстояние до боковых стен являлось значительно меньше расстояния до объекта. Так как в таком случае звук от боковых стен возвращался быстрее, чем от объекта, то выводимое расстояние было меньше фактического.

Также, в ходе тестирования был измерен фактический угол работы датчика. В исходном положении, датчик находился под углом  $0^\circ$  к нормали. Далее, датчик постепенно отклоняли вплоть до положения, в котором измеряемое датчиком расстояние перестало соответствовать реальному. Выяснилось, что максимальный угол, при котором датчик продолжает показывать расстояние до плоскости, к которой проведена нормаль, составляет около  $40^\circ$ , что является фактическим углом работы датчика. При этом производителем заявлен эффективный угол в  $15^\circ$ .

## **9 Вывод**

В ходе работы нами был разработан ультразвуковой дальномер с использованием Raspberry Pi и ультразвукового датчика HC-SR04. Для этого потребовалось спроектировать схему подключения, а также разработать программный код на языке Python для взаимодействия с датчиком по GPIO.

В ходе тестирования выяснилось, что проведение измерений в замкнутых пространствах невозможно из-за отражения звуковых волн от прочих стен помещения. Однако, в условиях близких к условиям открытого пространства, то есть когда расстояние до объекта меньше расстояния до прочих отражающих поверхностей, тесты показали удовлетворительный результат.

## 10 Исходные тексты программ

### 10.1 main.py

```
import time
import RPi.GPIO as GPIO

# -----
# Константы.
# -----


# Порты GPIO.
GPIO_TRIGGER = 23
GPIO_ECHO     = 24

# Остальные константы.
TIME_BETWEEN_MEASURE = 1 # Время между измерениями (в секундах).
SOUND_SPEED = 34300 # Скорость звука в воздухе.

# -----
# Нужные функции.
# -----


def measure():
    # Измерить расстояние.
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001) # Импульс должен быть 10 мкс, чтобы модуль выпустил сигнал.
    GPIO.output(GPIO_TRIGGER, False)
    start = time.time() # Время отправки сигнала.

    # Ждем, пока модуль не выставит ECHO в 1 (т.е. пока не отправит сигнал).
    while GPIO.input(GPIO_ECHO)==0:
        start = time.time()

    # Пока сигнал не пришел обратно - обновляем время прилета.
    while GPIO.input(GPIO_ECHO)==1:
        stop = time.time()

    pulse_duration = stop-start # Время прохождения импульса.
    distance = (pulse_duration * SOUND_SPEED)/2 # Вычисление расстояния.

    return distance

# Измерить среднее расстояние за 3 вычисления.
def measure_average():
    # Делаем 3 измерения и возвращаем среднее.
    distance1=measure()
    time.sleep(0.1)
    distance2=measure()
    time.sleep(0.1)
    distance3=measure()
    distance = distance1 + distance2 + distance3
    distance = distance / 3
    return distance

# -----
# Основная программа.
# -----


# Используем BCM номера (например, GPIO 23)
# вместо фактических порядковых номеров пинов.
GPIO.setmode(GPIO.BCM)
```

```
print("Измерение расстояния")

# Устанавливаем пины на вход и выход.
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)      # Trigger (издать звуковой сигнал).
GPIO.setup(GPIO_ECHO,GPIO.IN)         # Echo (сообщить, что приняли обратно сигнал).

# Устанавливаем Trigger в False (низкий уровень сигнала).
GPIO.output(GPIO_TRIGGER, False)

# Оборачиваем основные действия в блок try так,
# мы можем обработать нажатие пользователем "CTRL-C"
# и запустить функцию очистки GPIO. Кроме того, мы
# ограждаем пользователя от возможных сообщениях об ошибках.
try:

    while True:

        distance = measure_average()
        print ("Расстояние : %.1f сантиметров." % distance)
        time.sleep(TIME_BETWEEN_MEASURE)

except KeyboardInterrupt:
    # Пользователь нажал CTRL-C.
    # Сбрасываем настройки GPIO.
    GPIO.cleanup()
```

## **11 Список литературы и электронных источников**

- <https://vk.cc/ciHt99> - установка Linux на RaspberryPi.
- <https://vk.cc/ciHteo> - подключение модуля к RaspberryPi.
- <https://vk.cc/ciHtk8> - обработка данных с модуля.
- <https://vk.cc/ciHtqu> - подключение к RaspberryPi по SSH.