Лабораторная работа № 3.
“UART. Монитор порта. Библиотеки. Работа с сонаром и сервоприводом”

Цель работы

Изучение принципов работы последовательного интерфейса UART, использование порта UART для отладки программ.Изучение стандартных библиотек, а также библиотек необходимых для работы с сонаром и серводвигатель.

Задача

Ознакомиться с теоретическим материалом и выполнить задание по вариантам.

Необходимый материал для выполнения лабораторной работы

1. Arduino UNO
2. Breadboard
3. Сервопривод
4. Ультразвуковой дальномер
5. Монтажные провода
6. Светодиоды
7. Резисторы
8. Кнопки

Содержание отчёта

1. Цель работы.
2. Задание на работу по варианту.
3. Собранная схема подключения.
4. Блок-схема алгоритма.
5. Код, написанный на языке программирования С.
6. Выводы по работе.

Теоретическая часть

UART, монитор порта

1. UART

Универсальный асинхронный приёмопередатчик (Univsersal Asynchronos Reciever - Transmitter) - это физическое устройство приёма и передачи данных по двум проводам. Оно позволяет двум устройствам обмениваться данными на различных скоростях. В спецификацию UART не входят аналоговые уровни на которых ведётся общение между устройствами, UART это протокол передачи единиц и нулей. На практике UART часто называют последовательным портом.

У каждого устройства, поддерживающего UART обычно обозначены два вывода: **RX и TX**. TX — означает transmit (передаю), RX — receive (принимаю).

2. Методы связи

UART позволяет одновременно передавать и принимать данные, однако не всегда это возможно или нужно. Например, если Вам нужно только получать не критические данные (которые можно проверить следующим пакетом, например расстояние, посылаемое лидаром каждые несколько

сотен миллисекунд) от цифрового датчика или любого другого устройства и не нужно ничего передавать, такой метод называется симплексным. Всего различают три метода связи:

- **Полнодуплексная** — когда ведущий и ведомый могут одновременно принимать и передавать (одновременная передача в обе стороны)
- **Полудуплексная** — когда ведущий и ведомый поочерёдно принимают и передают (поочередная передача в обе стороны)
- **Симплексная** — когда ведущий или ведомый только передают (передача в одну сторону)

3. Инициализация последовательного порта

В скетче последовательный порт инициализируется функцией `Serial.begin(СКОРОСТЬ)` или `Serial.begin(СКОРОСТЬ, ПАРАМЕТРЫ)` в коде функции **setup**().

Пример:

```
void setup()
{
        Serial.begin(9600);
}
```

4. Скорость передачи данных

Скорость изменения логических уровней (импульсов) на линии принято измерять в **бодах**. Единица измерения названа так в честь французского изобретателя Жана Мориса Эмиля Бодо.

Бод (англ. baud) — единица измерения символьной скорости, количество изменений информационного параметра несущего периодического сигнала в секунду.

Скорость при использовании UART может быть любой, единственное требование — **скорости передающего и принимающего должны быть одинаковы**. Стандартная скорость UART принята за 9600 бод. Arduino без проблем и лишних настроек может принимать и передавать данные на скоростях до 115200 бод.

Так как при передаче данных присутствуют синхронизирующие биты, именуемые старт-бит и стоп-бит, не совсем корректно говорить, что скорость 9600 бод равна 9600 битам в секунду. Если речь идёт о полезных данных, то реальная скорость на 20% ниже. Например, если выставлены параметры 8-N-1 и 9600 бод, то на передачу одного байта уходит десять бит, и 9600/10 = 960 байт, что равно 7680 битам в секунду.

5. Основные функции

Serial.available();

- Вызов:

```
count = Serial.available();
```

- Описание:

Принимаемые по последовательному порту байты попадают в буфер микроконтроллера, откуда Ваша программа может их считать. Функция возвращает количество накопленных в буфере байт. Последовательный буфер может хранить до 128 байт.

- Возвращаемое значение:

Возвращает значение типа uint8_t (typedef uint8_t byte;) – количество байт, доступных для чтения, в последовательном буфере, или 0, если ничего не доступно.

- Пример:

```
if (Serial.available() > 0) {
  //Если в буфере есть данные, то будет выполнено тело условия
  //здесь должен быть прием и обработка данных
  }
```

Serial.read();

- Вызов:

```
char = Serial.read();
```

- Описание:

Считывает следующий байт из буфера последовательного порта.

- Возвращаемое значение:

Первый доступный байт входящих данных с последовательного порта, или (-1) если нет входящих данных.

- Пример:

```
incomingByte = Serial.read(); // читаем байт в incomingByte
```

Serial.readString ()

Функция readString() считывает символы из потока, объединяя их в строку.

- Пример:

```
String a = Serial.readString();
```

Serial.print()

Вывод данных на последовательный порт.

- Параметры:

Функция имеет несколько форм вызова в зависимости от типа и формата выводимых данных.

- Примеры:

```
Serial.print("Here our bytes:"); //выводит строку «Here our
//bytes:»

Serial.print(b); // если b имеет целый тип, выводит в порт
//десятичное представление числа b.
```

6. Структура UART - “сообщения”

Также нужно сказать, что данные обычно передаются блоками по 8 бит (1 байту) и сопровождаются стартовыми и стоповыми битами. Такие биты UART подставляет самостоятельно для удобства отслеживания правильности передачи данных, чтобы никакие данные не потерялись.

Однако есть варианты передачи данных не по 1 байту, а по 5, 6, 7, 9 бит. Но это зависит от настройки самого UART и цели использования такой конфигурации.

Кроме этого, некоторые конфигурации UART позволяют вставлять два стоповых бита. Это позволяет уменьшить рассинхрон передачи данных. Хотя приемник обычно игнорирует второй стоп, считая это за паузу.

Рисунок 1 - Структура пакета UART данных

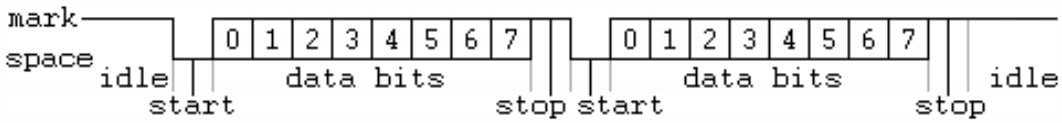

Рисунок 2 - Временная диаграмма передачи данных через UART

Пассивное состояние UART является логической единицей. В таком состоянии прием и передача данных не происходит (да и собственно никаких действий вообще не происходит). Поэтому стартовый бит является логическим нулем. Таким образом перепад с высокого уровня (1) на низкий (0) является знаком начала передачи при условии, что за время половины длительности бита всё ещё держится низкий уровень. Далее отсчитывается 9 битов (8 — данные, 1 последний — стоп). Причем значение стопового бита — это единица. Если UART получает что-то иное, то это считается за ошибку.

7. UART - связь между двумя платами

У каждого устройства, поддерживающего UART обычно обозначены два вывода: **RX и TX**. TX — означает transmit (передаю), RX — receive (принимаю). Отсюда становится понятно что для осуществления связи между двумя устройствами RX одного устройства нужно подключать к TX другого.

!!! RX - RX

Если Вы подключите RX одного устройства к RX другого, то оба устройства будут слушать друг друга, вы соедините входы устройств.

!!! TX - TX

Если соединить TX и TX - это уже более опасно, это выходы низкого сопротивления устройств и если на одном будет логическая единица, а на втором ноль — по проводу пойдёт ток короткого замыкания (это зависит от конкретной программной или аппаратной реализации). Хотя в современных чипах от этого есть защита, на всякий случай, не стоит на неё надеяться.

Так же необходимо объединить референсные уровни двух устройств (**GND-GND**), если не подразумевается гальваническая развязка. Пример соединения двух UNO приведен н рисунке 1.

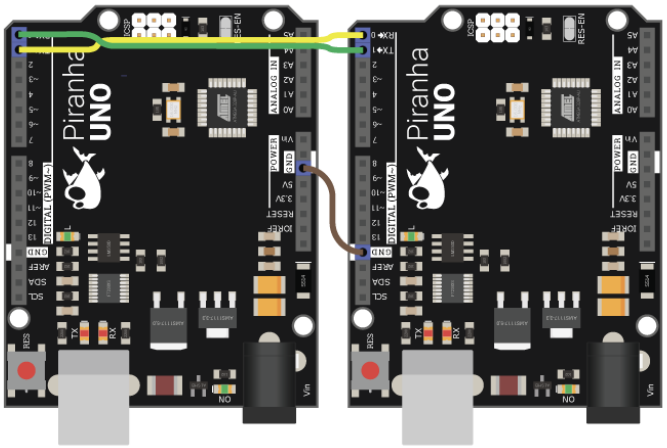

Рисунок 3 - Пример соединения двух UNO

Библиотеки. Сонар. Шаговый двигатель. Серводвигатель.

Возможности среды программирования Arduino могут быть существенно расширены за счет использования библиотек. Библиотеки расширяют функциональность программ и несут в себе дополнительные функции, например, для работы с аппаратными средствами, функции по обработке данных и т.д. Ряд библиотек устанавливается автоматически вместе со средой разработки, однако вы также можете скачивать или создавать собственные библиотеки.

С помощью библиотек можно выполнять сложные действия всего парой строк кода, потому что кто-то другой уже написал часть кода за вас.

В Arduino IDE удобный интерфейс для работы с библиотеками ардуино. Прямо из меню программы вы можете скачать, установить и подключить в свой скетч множество библиотек. Для большинства библиотек Arduino можно посмотреть примеры использования. Это поможет понять принцип работы библиотеки. Примеры можно доработать под свои нужды и использовать для реализации своих устройств.

Существуют стандартные библиотеки, которые устанавливаются вместе с Arduino IDE. Некоторые из них даже автоматически

подключаются в скетч (например Serial, с этой встроенной библиотекой мы уже работали).

В рамках данной работы вам предстоит познакомиться с такими библиотеками как: **Servo, Stepper.**

Библиотека Servo

Arduino библиотека Servo представляет собой набор функций для управления сервоприводами[1]. Данная библиотека автоматически устанавливается вместе с Arduino IDE. Стандартные серводвигатели позволяют задавать угол поворота вала в диапазоне от 0 до 180 градусов. Схема для подключения сервопривода приведена на рисунке 1.

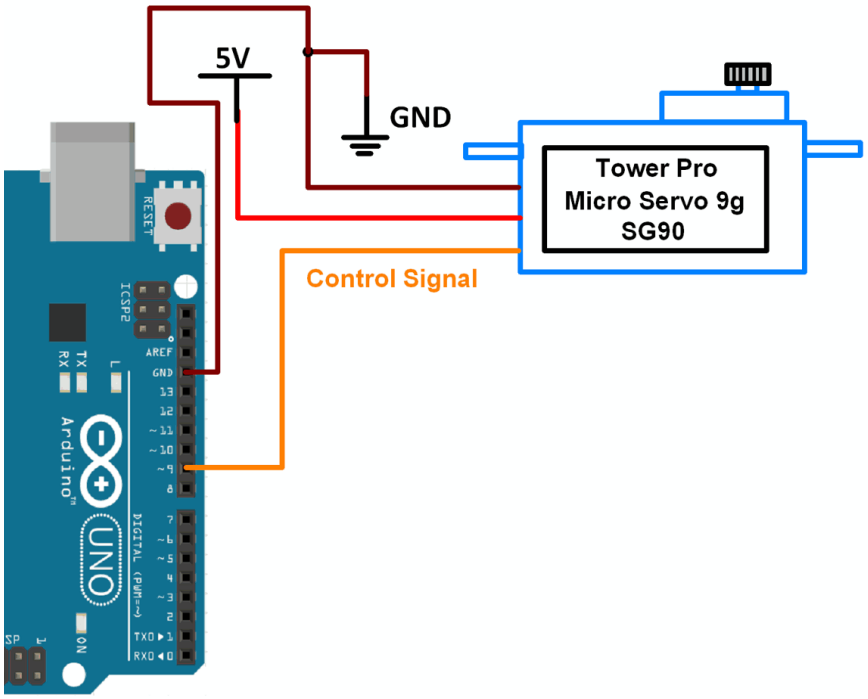

Рисунок 1 - Подключение сервопривода

Для использования библиотеки Servo необходимо подключить ее в свой скетч и создать переменную servo. Сделать это очень просто:

[1] Сервопривод представляет собой двигатель со встроенным редуктором и выходным валом, положение которого можно точно контролировать.

```
#include <Servo.h> //Подключение библиотеки
Servo myservo;//создаем переменную типа Servo

void setup() {
 // CODE...
}

void loop() {
 // CODE...
}
```

Далее необходимо указать пин к которому подключен сервопривод, это можно сделать с помощью функции `attach()`, применив её к созданному объекту `myservo.attach(pin, min, max)`, где

pin — Обязательный параметр. Цифровой пин к которому подключен сигнальный провод сервопривода.

min — Необязательный параметр. Ширина импульса в микросекундах, соответствующая минимальному (угол 0 градусов) положению сервопривода. (по умолчанию 544)

max — Необязательный параметр. Ширина импульса в микросекундах, соответствующая максимальному (угол 180 градусов) положению сервопривода.

```
#include <Servo.h>

Servo myservo;

void setup() {
  myservo.attach(9);
}

void loop() {
  // CODE...
}
```

После того как создан объект и указан пин, к которому подключен сервопривод, можно использовать функции для управления приводом.

Функция `write()` поворачивает сервопривод на заданный угол. Для сервоприводов постоянного вращения устанавливает скорость и направление вращения `myservo.write(angle)`, где

angle — обязательный параметр, в обычных сервоприводах отправляемое значение задает угол поворота вала в градусах. В сервоприводах непрерывного вращения отправляемое значение задает скорость вращения вала (0 - максимальная скорость в одном направлении, 180 - максимальная скорость в другом направлении; отправка значения, приблизительно равного 90, приводит к остановке двигателя).

```
#include <Servo.h>

Servo myservo;

void setup() {
 myservo.attach(9);
 myservo.write(90); // Поворачивает сервопривод на
                    //среднее положение
}

void loop() {
 // CODE...
}
```

Функция `read()` возвращает текущее положение сервопривода.

Возвращаемые значения: Int от 0 до 180.

```
#include <Servo.h>

Servo myservo;

void setup() {
 Serial.begin(9600); //Открываем последовательный порт
 myservo.attach(9);
 int position = myservo.read(); // Считываем положение
```

```
                                        //сервопривода
  Serial.print("Текущее положение сервопривода: ");
  Serial.println(position); // Отправляем значение угла
                     //на запись в последовательный порт
}

void loop() {
  // CODE...
}
```

Библиотека Stepper

Данная библиотека предназначена для работы с шаговыми двигателями[2] Так же как и сервоприводы, шаговые моторы являются крайне важным элементом автоматизированных систем и робототехники. Их можно найти во многих устройствах рядом: от CD-привода до 3D-принтера или робота-манипулятора.

Мы будем работать с двигателем модели 28BYJ-48. Это униполярный шаговый двигатель. Используется в небольших проектах роботов, сервоприводных устройствах, радиоуправляемых приборах.

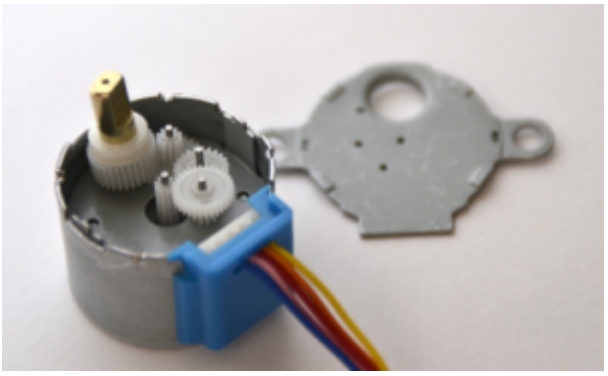

Рисунок 2 - Шаговый двигатель 28BYJ-48

[2] Шаговый двигатель – это мотор, перемещающий свой вал в зависимости от заданных в программе микроконтроллера шагов и направления. Подобные устройства чаще всего используются в робототехнике, принтерах, манипуляторах, различных станках и прочих электронных приборах. Большим преимуществом шаговых двигателей над двигателями постоянного вращения является обеспечение точного углового позиционирования ротора. Также в шаговых двигателях имеется возможность быстрого старта, остановки, реверса.

Характеристики двигателя:

- Номинальное питание – 5В;
- 4-х фазный двигатель, 5 проводов;
- Число шагов: 64;
- Угол шага 5,625°;
- Скорость вращения: 15 оборотов в секунду
- Крутящий момент 450 г/сантиметр;
- Сопротивление постоянного тока 50Ω ± 7% (25 ℃).

Драйвер – это устройство, которое связывает контроллер и шаговый двигатель. Для управления биполярным шаговым двигателем чаще всего используется драйверы L298N и ULN2003, мы будем использовать второй.

Имеем униполярный двигатель 28BYj-48 и драйвер ULN2003. Схема подключения двигателя и драйвера приведена на рисунке 2.

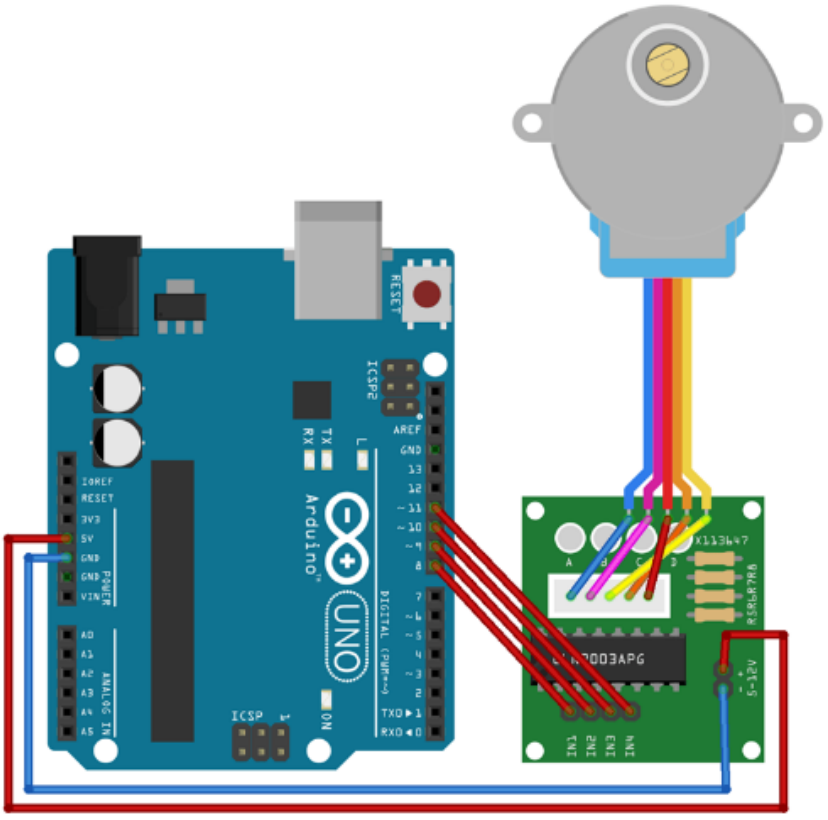

Рисунок 3 - Схема подключения шагового двигателя и драйвера

Для работы с шаговым двигателем сначала необходимо подключить соответствующую библиотеку, а также создать экземпляр класса Stepper, с помощью функции с аналогичным именем `Stepper(steps, pin1, pin2, pin3, pin4)`.

steps: количество шагов, необходимых для того, чтобы вал двигателя совершил полный оборот. Если в документации к двигателю указано, сколько градусов составляет один шаг, то для расчета параметра steps необходимо разделить 360 на показатель в документации (т.е. 360 / 3.6 = 100 шагов). (int)

pin1, pin2: два вывода, к которым подсоединен шаговый двигатель (int)

pin3, pin4 (не обязательные параметры): последние два вывода, к которым подсоединен шаговый двигатель при четырехпроводной схеме подключения (int).

Данную функцию необходимо использовать в начале программы, до вызова setup() и loop(). Количество параметров, передаваемых функции Stepper, зависит от схемы подключения двигателя к Ардуино - по двухпроводной схеме или по четырехпроводной соответственно.

```
#include <Stepper.h> //подключение библиотеки

Stepper myStepper = Stepper(100, 5, 6, 7, 8);
//создание экземпляра двигателя, подключенного к пинам
//5-8, полный оборот которого состоит из 100 шагов

void setup() {}
void loop() {}
```

Функция `setSpeed(rpms)` задает скорость вращения двигателя в оборотах в минуту (RPM). Данная функция не вызывает движения вала

двигателя, а лишь задает его скорость вращения после вызова функции step().

rpms: скорость вращения шагового двигателя в оборотах в минуту - положительное число (long)

Скорость вращения обычно задается в **setup()**.

Функция `step(steps)` задает поворот вала двигателя на указанное количество шагов.

steps: количество шагов, на которое следует повернуть вал двигателя - положительное число приводит к вращению в одну сторону, отрицательное - в другую (int)

При этом поворот осуществляется на скорости, заданной последним вызовом функции setSpeed(). **Данная функция является блокирующей.** Это значит, что выполнение программы не перейдет на следующую строку до тех пор, пока не завершится работа функции step(). Например, если для шагового двигателя, имеющего 100 шагов, вы установили скорость вращения, скажем, 1 RPM (один оборот в минуту), то выполнение функции step(100) займет одну минуту. Во избежание этого лучше задавать высокую скорость вращения и стараться вызывать функцию step() с небольшим количеством шагов.

Библиотека Ultrasonic

Ультразвуковой датчик определяет расстояние до объекта так же, как это делают летучие мыши или дельфины. Датчик HC-SR04 генерирует узконаправленный сигнал на частоте 40 кГц и ловит отраженный сигнал (эхо). По времени распространения звука до объекта и обратно можно достаточно точно определить расстояние до него.

По этому же принципу работает множество приборов для исследования пространства — эхолот, сонар, радиолокатор и даже полицейский радар для определения скорости автомобиля. Все эти приборы излучают узконаправленный ультразвуковой сигнал и получают обратно отраженный сигнал. В отличии от инфракрасных дальномеров (IR), на показания ультразвукового датчика не влияет цвет объекта.

Но при настройке ультразвукового датчика на Ардуино могут возникнуть трудности с определением расстояния до звукопоглощающих объектов, поскольку они способны полностью погасить излучаемый сигнал. Для идеальной точности измерения расстояния, поверхность изучаемого объекта должна быть ровной и гладкой. Принцип работы ультразвукового датчика hc-sr04 показан на рисунке 4.

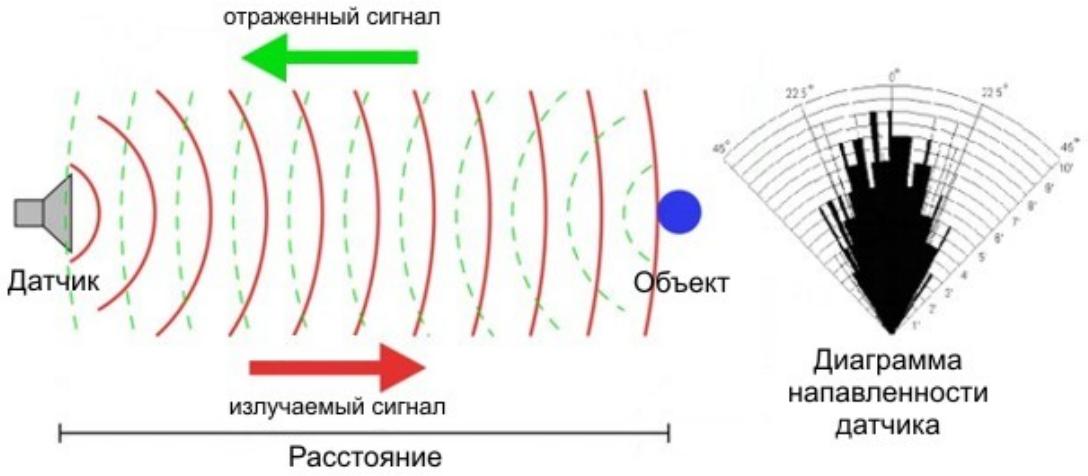

Рисунок 4 - Принцип работы ультразвукового датчика

Схема подключения указана на рисунке 5. Отметим, что ультразвуковой дальномер HC-SR04 имеет диапазон измерения от 2 см до 400 см, работает при температурах от 0° до 60° С. Точность измерения составляет ± 1 см, рабочее напряжение датчика до 5,5 В.

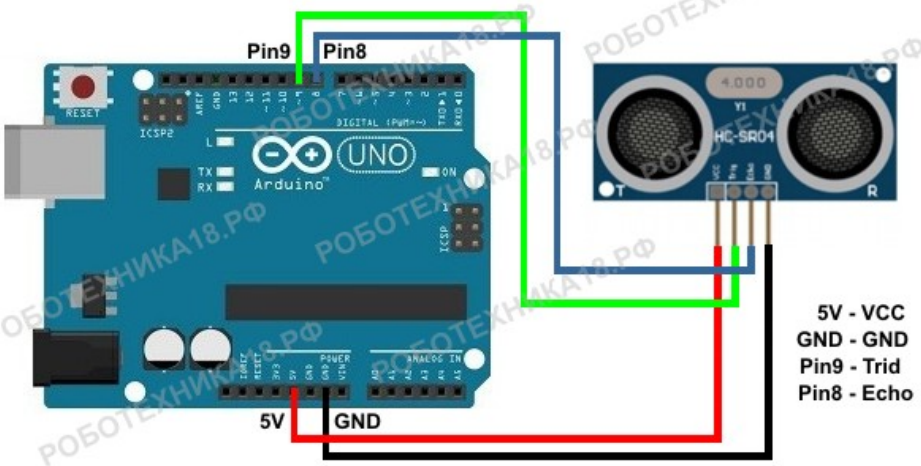

Рисунок 5 - Схема подключения дальномера

Если библиотека **Ultrasonic** не обнаруживается, необходимо ее догрузить через Arduino IDE.

Ниже приведен код, обеспечивающий вывод показаний датчика в монитор порта.

```
   #include <Ultrasonic.h> //подключаем библиотеку для работы с сонаром

   Ultrasonic ultrasonic(12, 13);//(trig, echo)// создаем экземпляр класса
   int distance; //создаем переменную для расстояния

   void setup() {
    Serial.begin(9600);//инициируем последовательный порт
   }

   void loop() {
        distance = ultrasonic.read(); // считываем расстояние и помещаем значение в переменную distans

     Serial.print("Distance in CM: ");
     Serial.println(distance);
     delay(1000); // с помощью задержки задаем периодичность измерений
   }
```

Задания

Первый вариант

1.1 Реализовать управление работой 3-х светодиодов через последовательный порт. Первая команда, отправленная в порт должна указывать на порядковый номер светодиода, вторая команда должна означать **вкл** или **выкл** соответствующего светодиода.

1.2 Реализовать сигнализатор расстояния до препятствия с помощью ультразвукового дальномера и RGB светодиода. При расстоянии более 30 см светодиод должен гореть зеленым светом, при расстоянии от 10 см до 30 см - желтым, при расстоянии менее 10 см - красным.

Второй вариант

2.1 Придумайте три светодиодных эффекта, переключение между которыми можно осуществлять при отправке различных символов с ПК. Светодиоды должны быть подключены в один ряд, “гирлянда” должна состоять из 5 светодиодов.

2.2 Реализовать сигнализатор расстояния до препятствия с помощью ультразвукового дальномера и трех светодиодов. При расстоянии более 30см должен гореть один светодиод, при расстоянии от 10см до 30см - два, при расстоянии менее 10см - три.

Третий вариант

3.1 Реализовать управление яркостью гирлянды из 3-х светодиодов через последовательный порт. Должно быть реализовано 3 режима свечения разной яркости. Изначально все светодиоды выключены.

3.2 Реализовать регулятор яркости светодиода, в зависимости от показаний ультразвукового датчика. Чем ближе препятствие, тем ярче горит светодиод.

Четвертый вариант

4.1 Реализуйте управление дорожкой из светодиодов. Включение/выключение гирлянды должно осуществляться согласно соответствующей команде, поступающей с ПК. Светодиоды должны быть подключены в один ряд, “гирлянда” должна состоять из 5 светодиодов.

4.2 Реализовать измерение расстояния до препятствия по нажатию кнопки, и вывод полученных значений в последовательный порт.

Пятый вариант

5.1 Реализовать управление работой 5-ти светодиодов через последовательный порт. Команда, отправленная в порт должна указывать на порядковый номер светодиода, который в свою очередь должен сменить состояние (если был включен, то выключиться, если выключен, то включиться). Изначально все светодиоды выключены.

5.2 Реализовать управление положением сервопривода с помощью потенциометра, с выводом угла вала сервопривода в последовательный порт.

Шестой вариант

6.1 Напишите программу, мигающую светодиодами с частотой в соответствии с режимом, заданным с ПК. Светодиоды должны быть подключены в один ряд, “гирлянда” должна состоять из 5 светодиодов.

6.2 Реализовать управление валом сервопривода через последовательный порт. В последовательный порт отправляется значение угла - вал сервопривода принимает соответствующее положение.

Седьмой вариант

6.1 С помощью команд отправленных с компьютера менять направление изменения значений с потенциометра и номер светодиода у которого будет

меняться яркость светодиода, который, в свою очередь, управляет яркостью двух светодиодов. Команды по примеру: “one” - яркость меняется на первом светодиоде, “two” - яркость меняется на втором светодиоде, “direct” - меняется направление изменения яркости.

6.2 Реализовать управление валом сервопривода с помощью вращения ручки потенциометра, вал сервопривода повторяет движение ручки потенциометра.

Полезные ссылки

1. https://wiki.iarduino.ru/page/serial-protocols-uart/ - UART - Универсальный Асинхронный Приёмопередатчик
2. https://www.arduino.cc/reference/en/language/functions/communication/serial/ - здесь можно найти описание основных функций UART
3. **https://doc.arduino.ua/ru/prog/ - Справочник по работе с Ардуино**
4. https://doc.arduino.ua/ru/prog/Libraries - Основные библиотеки Ардуино