Практическое занятие №6
Тема: «Подключение и программирование ультразвукового датчика HC-SR04»

Цель работы: приобрести практические навыки по подключению и программированию кнопочных массивов и многоцветных светодиодов на платформе Arduino.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно заданию в примере.

Содержание отчета:

– название практического занятия, его цель;
– фото или скриншоты собранной схемы;
– написанный программный код вставить текстом;
– вывод о проделанной работе;
– файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Принцип работы

HC-SR04 – это ультразвуковой датчик расстояния, который работает по принципу эхолокации. Датчик генерирует высокочастотные звуковые волны (40 кГц), волны отражаются от объектов и возвращаются к датчику, и полученные данные вычисляется время между отправкой и приемом сигнала.

Технические характеристики

1. Рабочее напряжение**:** 5V DC
2. Рабочий ток**:** 15 мА
3. Частота ультразвука**:** 40 кГц
4. Измерительный диапазон**:** 2 см - 400 см
5. Точность**:** ±3 мм
6. Угол обзора**:** 15 градусов
7. Габариты: 45×20×15 мм

Распиновка датчика

Пин	Назначение
VCC	Питание +5V
Trig	Запуск измерения (Input)
Echo	Получение результата (Output)
GND	Земля

Физический принцип

Скорость звука в воздухе при нормальных условиях составляет примерно: 343 м/с при 20°C, 331 м/с при 0°C.

Формула расчета расстояния:

Расстояние = (Время × Скорость_звука) / 2 — *деление на 2 учитывает, что звук проходит путь до объекта и обратно.*

Протокол работы

1. Инициализация измерения**:**
- на Trig подается импульс длительностью 10 мкс
- датчик автоматически отправляет 8 импульсов 40 кГц

2. Получение результата**:**
- на Echo устанавливается высокий уровень
- длительность высокого уровня пропорциональна расстоянию

Особенности и ограничения

Преимущества:
- низкая стоимость
- простота подключения и программирования
- достаточная точность для большинства проектов
- не подвержен влиянию света и цвета объектов

Ограничения:
- плохо работает с мягкими и пористыми материалами (поглощают звук)
- может давать ошибки при измерениях под углом
- чувствителен к температуре и влажности воздуха
- не может измерять очень близкие объекты (<2 см)

Корректировка на температуру
Для повышения точности нужно учитывать температуру:

```
// Пример корректировки на температуру
float temperature = 20.0; // °C
float speedOfSound = 331.3 + 0.606 * temperature; // м/с
distance = (duration * speedOfSound * 0.0001) / 2; // в см
```

Типичные применения данного датчика

– системы парковки автомобилей
– беспилотные транспортные средства
– робототехника (обнаружение препятствий)
– измерительные системы
– системы безопасности

Рисунок 1 – Принцип подключения RGB-светодиода

ЗАДАНИЕ

Построить на макетной плате и нарисовать схему во Fritzing:

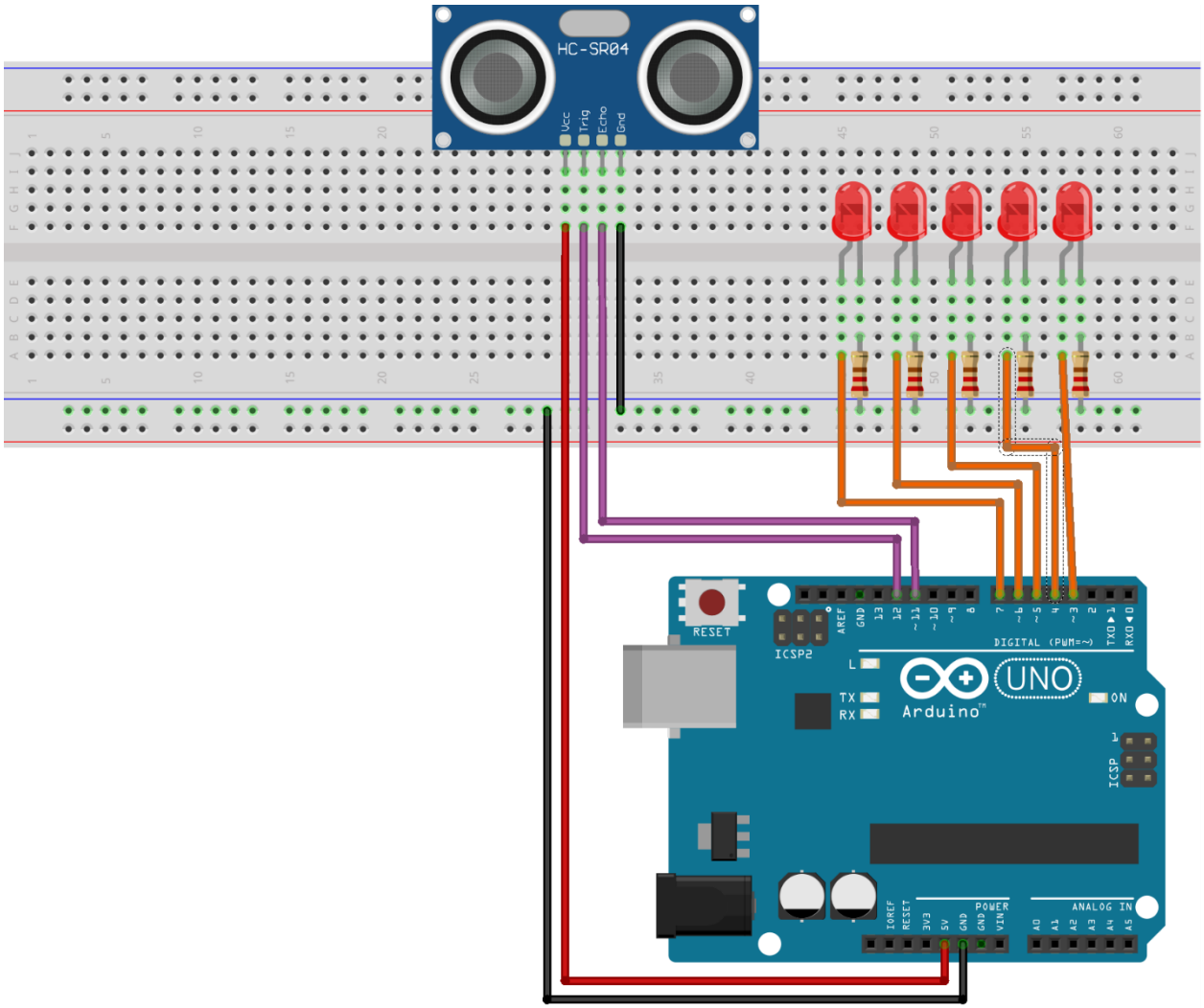

Рисунок 2 – Схема кодключения

Написать программу:

```
// Определение пинов для ультразвукового датчика
const int trigPin = 12;
const int echoPin = 11;

// Пins для светодиодов
const int leds[] = {3, 4, 5, 6, 7};
const int numLeds = 5;

// Переменная для хранения предыдущего состояния
int previousLevel = -1;
```

```
void setup() {
  // Инициализация последовательного порта
  Serial.begin(9600);

  // Настройка пинов датчика
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Настройка пинов светодиодов
  for (int i = 0; i < numLeds; i++) {
    pinMode(leds[i], OUTPUT);
    digitalWrite(leds[i], LOW);
  }
}

void loop() {
  // Измерение расстояния
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration / 58; // Перевод в сантиметры

  // Вывод расстояния в монитор порта
  Serial.print("Расстояние: ");
  Serial.print(distance);
  Serial.println(" см");

  // Определение текущего уровня (каждые 10 см)
  int currentLevel = -1;
  if (distance <= 50) {
    currentLevel = distance / 10; // 0-10 см = уровень 0,
10-20 = уровень 1 и т.д.
  }

  // Если уровень изменился в большую сторону
  if (currentLevel > previousLevel) {
    // Зажигаем все светодиоды до текущего уровня
    for (int i = 0; i <= currentLevel && i < numLeds; i++) {
```

```
      digitalWrite(leds[i], HIGH);
    }
    previousLevel = currentLevel;
  }
  // Если уровень уменьшился
  else if (currentLevel < previousLevel) {
    // Гасим светодиоды выше текущего уровня
    for (int i = currentLevel + 1; i <= previousLevel && i <
numLeds; i++) {
      digitalWrite(leds[i], LOW);
    }
    previousLevel = currentLevel;
  }

  delay(1000); // Задержка между измерениями
}
```