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

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:

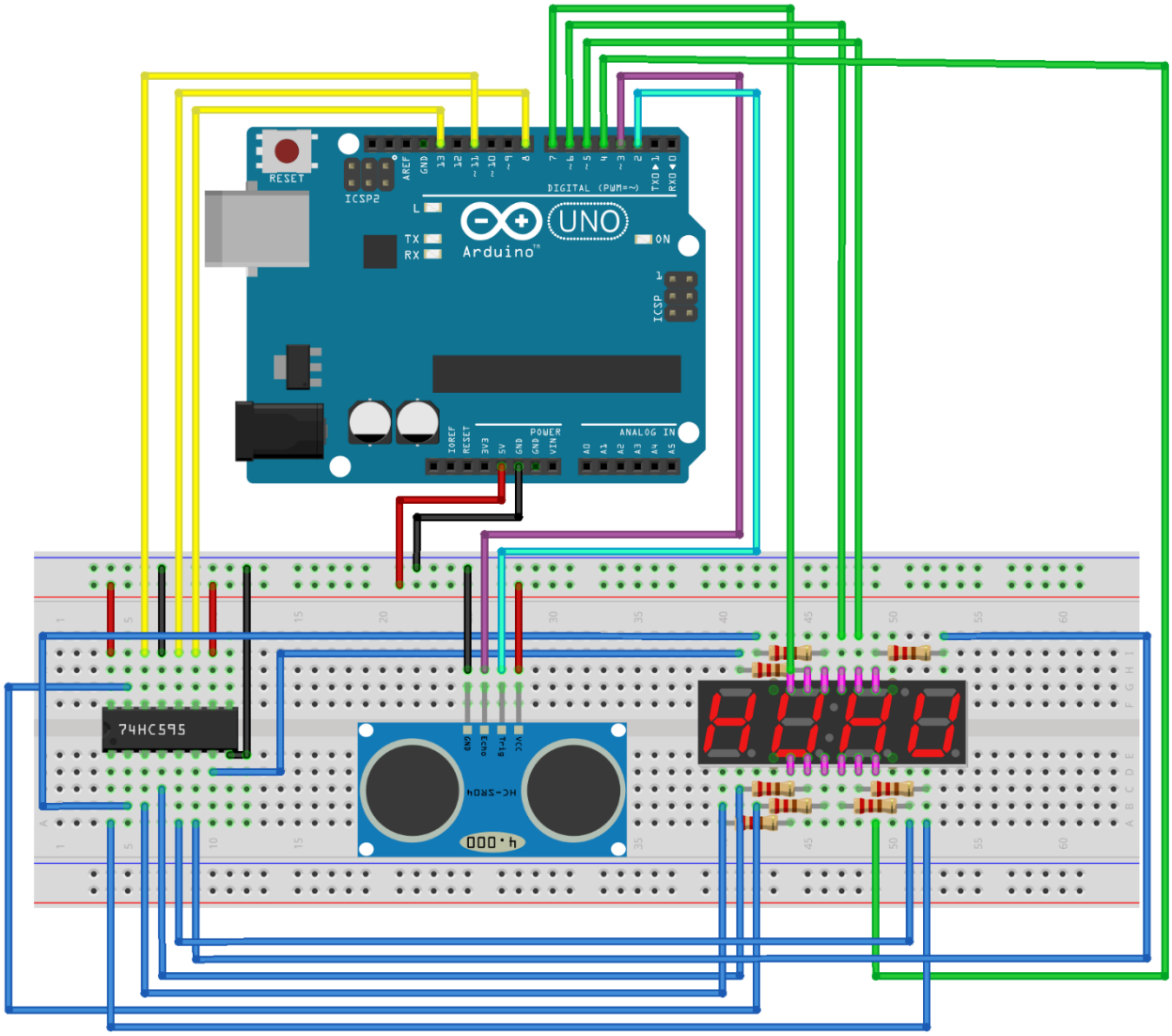

Рисунок 2 – Схема кодключения

Написать программу:

```
#include <SPI.h>

// Пин для latch (SS)
const int pin_spi_ss = 8;

// Пины для HC-SR04
const int trigPin = 2;
const int echoPin = 3;

byte numbers[10] = {
  //GFEDCBA
  B00000011, // 0
  B10011111, // 1
  B00100101, // 2
  B00001101, // 3
  B10011001, // 4
  B01001001, // 5
  B01000001, // 6
  B00011111, // 7
  B00000001, // 8
  B00001001, // 9
};

// Пины для управления разрядами индикатора (катоды)
int pindigits[4] = {7, 6, 5, 4};

// Переменные для хранения расстояния
float distance_cm = 0;
int cm_value = 0;
int mm_value = 0;
unsigned long lastMeasurement = 0;

void setup() {
  Serial.begin(9600);
  SPI.begin();

  // Настройка пина latch как выхода
  pinMode(pin_spi_ss, OUTPUT);
  digitalWrite(pin_spi_ss, HIGH);

  // Настройка пинов разрядов как выходов
  for (int i = 0; i < 4; i++) {
    pinMode(pindigits[i], OUTPUT);
    digitalWrite(pindigits[i], LOW);
  }

  // Настройка пинов для HC-SR04
  pinMode(trigPin, OUTPUT);
```

```
  pinMode(echoPin, INPUT);

  Serial.println("Инициализация завершена");
  Serial.println("Измерение расстояния...");
}

void loop() {
  // Измеряем расстояние раз в секунду
  if (millis() - lastMeasurement >= 1000) {
    lastMeasurement = millis();
    measureDistance();

    // Выводим значение в консоль
    Serial.print("Расстояние: ");
    Serial.print(distance_cm);
    Serial.println(" см");
  }

  // Отображаем расстояние на индикаторе
  displayDistance();
}

void measureDistance() {
  // Генерируем импульс 10 мкс
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Измеряем длительность импульса
  long duration = pulseIn(echoPin, HIGH);

  // Рассчитываем расстояние в см
  distance_cm = duration * 0.034 / 2;

  // Ограничиваем максимальное расстояние 999.9 см
  if (distance_cm > 999.9) distance_cm = 999.9;
  if (distance_cm < 0) distance_cm = 0;

  // Разделяем на сантиметры и миллиметры
  cm_value = (int)distance_cm;          // целая часть - сантиметры
  mm_value = (int)((distance_cm - cm_value) * 10); // дробная часть - миллиметры
}

void displayDistance() {
  int digits[4] = {0};

  // Заполняем массив цифр для отображения
  digits[0] = cm_value / 100;                    // сотни сантиметров
  digits[1] = (cm_value / 10) % 10;              // десятки сантиметров
```

```
  digits[2] = cm_value % 10;                    // единицы сантиметров
  digits[3] = mm_value;                         // миллиметры

  // Отображаем цифры на индикаторе
  for (int i = 0; i < 4; i++) {
    // Выключаем все разряды
    for (int j = 0; j < 4; j++) {
      digitalWrite(pindigits[j], LOW);
    }

    // Включаем текущий разряд
    digitalWrite(pindigits[i], HIGH);

    // Для второго разряда (единицы см) добавляем точку
    if (i == 2) {
      showNumberWithDot(digits[i]);
    } else {
      showNumber(digits[i]);
    }

    // Короткая задержка для динамической индикации
    delay(2);
  }
}

// Функция вывода цифры без точки
void showNumber(int num) {
  digitalWrite(pin_spi_ss, LOW);
  SPI.transfer(numbers[num]);
  digitalWrite(pin_spi_ss, HIGH);
}

// Функция вывода цифры с точкой
void showNumberWithDot(int num) {
  digitalWrite(pin_spi_ss, LOW);
  // Сбрасываем старший бит для включения точки (DP)
  SPI.transfer(numbers[num] & B11111110);
  digitalWrite(pin_spi_ss, HIGH);
}
```