Базовые функции микроконтроллера на базе Arduino (продолжение)

13 лекция

«Программирование микроконтроллеров на языке C++»

Генераторы случайных значений

Функция **random()** возвращает псевдослучайное число.

Синтаксис

- random(max)
- random(min, max)

Параметры

- **min -** нижняя граница случайных значений, включительно. (опционально)
- **max** - верхняя граница случайных значений, не включительно.

Возвращаемое значение

- целое число типа **long** в диапазоне **от min до (max − 1)**

Пример использования функции random()

```
long randNumber;

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

/* если порт 0 не подключен, то генератор псевдослучайных чисел будет
инициализироваться функцией randomSeed() со случайного значения при каждом
запуске программы из-за "шума" на порту */
  randomSeed(analogRead(0)); }

void loop() {
 // выводим случайное число из диапазона 0..299
  randNumber = random(300);
  Serial.println(randNumber);
   // выводим случайное число из диапазона 10..19
  randNumber = random(10, 20);
  Serial.println(randNumber);
   delay(50); }
```

Генераторы случайных значений

Функция **randomSeed(seed)**

seed - параметр, задающий начало выдачи псевдослучайных значений на последовательности (int, long)

Возвращаемое значение – нет.

- Функция **randomSeed()** инициализирует генератор псевдослучайных чисел. Генерируемая последовательность случайных чисел очень длинная, и всегда одна и та же. Точка в этой последовательности, с которой начинается генерация чисел, зависит от параметра **seed**.
- Если при каждом запуске программы необходимо получать разные последовательности значений, генерируемых функцией **random()**, то необходимо инициализировать генератор псевдослучайных чисел со случайным параметром. Например, можно использовать значение, отдаваемое функцией **analogRead()** с неподключенного порта вход/выхода.

Пример использования функции randomSeed(seed)

Пример. Программа каждые 50 мс генерирует новое случайное число от 0 до 299 и выводит его в Serial Monitor. Начальная энтропия берётся из шумового значения аналогового входа, чтобы последовательность была разной при каждом запуске.

```
long randNumber;

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop() {
  randNumber = random(300);
  Serial.println(randNumber);
   delay(50);
}
```

Пример использования функции randomSeed(seed)

В некоторых случаях необходимо получать одинаковую последовательность при каждом запуске программы на Arduino. В этом случае инициализировать генератор псевдослучайных чисел следует вызовом функции **randomSeed()** с фиксированным параметром.

```
long randNumber;

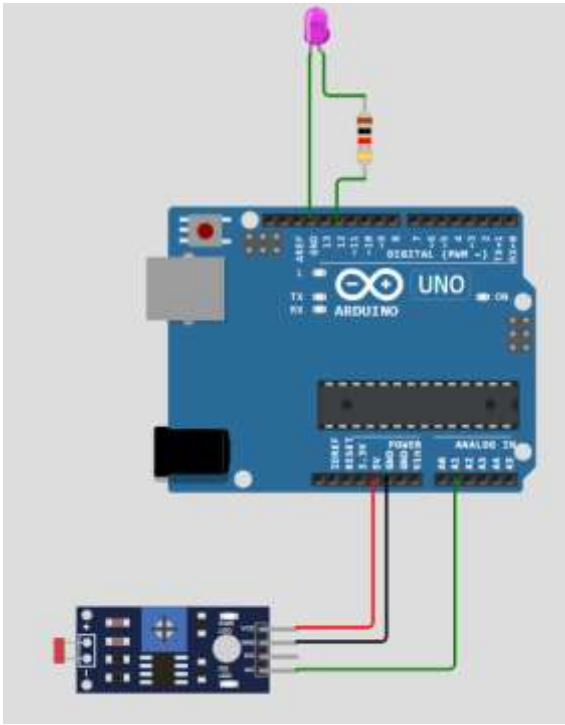
void setup() {
  Serial.begin(9600);
    randomSeed(1234); // Фиксированное начальное значение
     генератора
}
void loop() {
  randNumber = random(100);   // числа от 0 до 99
  Serial.println(randNumber);
  delay(500);
}
```

```
int sensorPin = A1;    // фоторезистор подключён к A1
int led1 = 12;             // светодиод
const int NOISE_MIN = -50; // Параметры шума
const int NOISE_MAX =  50; // Параметры шум

void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  randomSeed(analogRead(0));  // инициализация случайных чисел
 }
void loop() {
      // 1. Считывание реального значения ADC
  int adcReal = analogRead(sensorPin);

      // 2. Случайная добавка
  int randomDelta = random(NOISE_MIN, NOISE_MAX + 1);

       // 3. Итоговое значение
  int adcWithNoise = adcReal + randomDelta;
  adcWithNoise = constrain(adcWithNoise, 0, 1023);
…….
}
```



Функции передачи данных

- Набор функций **Serial** служит для связи устройства Ардуино с компьютером или другими устройствами, поддерживающими последовательный интерфейс обмена данными. Все платы Arduino имеют хотя бы один последовательный порт UART.
- Для обмена данными **Serial** используют цифровые порты ввод/вывода 0 (RX) и 1 (TX), а также USB порт. Важно учитывать, что если вы используете функции **Serial**, то нельзя одновременно с этим использовать порты 0 и 1 для других целей как обычные цифровые пины.
- Среда разработки Arduino имеет встроенный монитор последовательного интерфейса Serial monitor. Для начала обмена данными необходимо запустить монитор нажатием кнопки Serial monitor и выставить ту же скорость связи (baud rate), с которой вызвана функция begin().

Таблица функций Serial

Функция	Синтаксис	Назначение	Пример
begin()	Serial.begin(speed);	Инициализация последовательного порта и установка скорости передачи данных	Serial.begin(9600);
end()	Serial.end();	Отключение последовательного порта	Serial.end();
available()	Serial.available();	Возвращает количество байт, доступных для чтения	if (Serial.available() > 0)
read()	Serial.read();	Читает один байт из входного буфера	char c = Serial.read();
flush()	Serial.flush();	Ожидает завершения передачи всех данных	Serial.flush();

Таблица функций Serial

Функция	Синтаксис	Назначение	Пример
print()	Serial.print(data);	Передаёт данные без перехода на новую строку	Serial.print("Temp: ");
println()	Serial.println(data);	Передаёт данные и добавляет перевод строки	Serial.println(25);
write()	Serial.write(байт/массив);	Отправляет байт или массив байт в «сыром» виде	Serial.write(65); // отправит 'A'
peek()	Serial.peek();	Просматривает следующий байт, не удаляя его из буфера	char c = Serial.peek();

Заключение

- В ходе двух лекций были рассмотрены ключевые элементы программирования микроконтроллеров Arduino.
- Базовые функции цифрового и аналогового ввода/вывода позволяют управлять внешними компонентами и получать информацию от датчиков.
- Функции работы со временем обеспечивают организацию задержек и измерение интервалов в программах.
- Набор функций Serial предоставляет возможность обмена данными между платой и компьютером или внешними устройствами, поддерживающими UART-интерфейс.
- Совместное использование этих инструментов формирует основу для создания интерактивных, управляемых и интеллектуальных систем, позволяя проектировать сложные приложения на базе микроконтроллеров.