

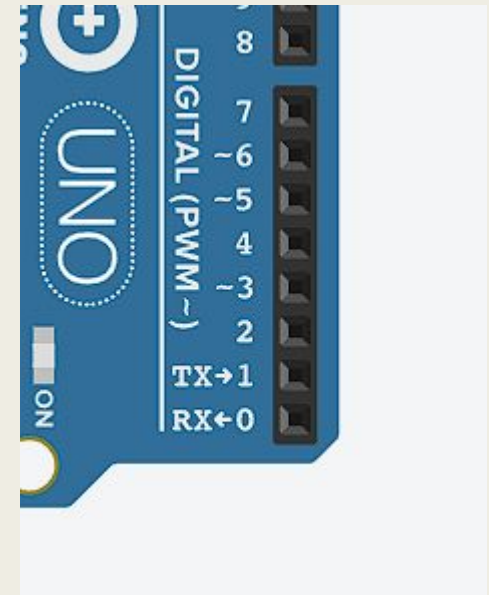
Основы кибернетики и робототехники

Лекция 5

Serial. Общаемся с компьютером

Набор функций **Serial** — это стандартный набор функций, который используется для передачи данных через последовательный порт **Arduino**. Последовательный порт работает с двумя цифровыми пинами Ардуино 0-ой (RX) и 1-ый (TX). В большинстве плат ардуино доступен 1 интерфейс Serial.

Среда Arduino IDE не содержит отладчика, что создает определенные проблемы в поиске ошибок кода программы. Без ошибок программы сразу не пишутся. Формальные ошибки выявляются при компиляции, а с алгоритмическими и вычислительными ошибками намного сложнее. Основная функция отладки это увидеть состояние программы, узнать значение переменных. Это можно сделать, передав нужную информацию на компьютер через последовательный интерфейс. Физическое подключение платы Ардуино к компьютеру через USB кабель существует всегда. Среда Arduino IDE имеет монитор последовательного порта, позволяющий получать и посылать данные обмена с платой. Можно передать на компьютер любую информацию о состоянии



Последовательный интерфейс UART.

UART в переводе это универсальный асинхронный приемопередатчик. Данные UART передаются последовательным кодом в следующем формате.



Каждый бит передается за равные промежутки времени. Время передачи одного бита определяется скоростью передачи.

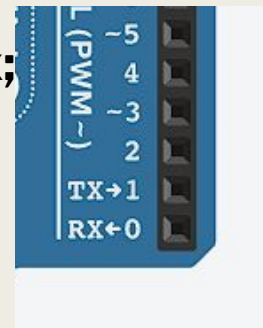
Часто используются следующие стандартные скорости передачи интерфейса UART.

| Скорость передачи, бод | Время передачи одного бита, мкс | Время передачи байта, мкс |
|---------------------------|------------------------------------|------------------------------|
| 4800 | 208 | 2083 |
| 9600 | 104 | 1042 |
| 19200 | 52 | 521 |
| 38400 | 26 | 260 |
| 57600 | 17 | 174 |
| 115200 | 8,7 | 87 |

Бод (англ. baud) в связи и электронике — единица измерения символьной скорости.

Обмен информацией через UART происходит в двойном режиме, т.е. передача данных может происходить одновременно с приемом. **Для этого в интерфейсе UART есть два сигнала:**

TX – выход для передачи данных;
RX – вход для приема данных.



При соединении двух UART устройств выход TX одного устройства соединяется со входом RX другого. А сигнал TX второго UART подключается к входу RX первого.

Библиотека **Serial** для работы с **UART**

Ардуино.

Для работы с аппаратными UART контроллерами в Ардуино существует встроенный класс **Serial**. Он предназначен для управления обменом данными через UART. Перед тем, как перейти к функциям класса **Serial**, необходимо понять разницу в формате данных обмена.

Через последовательный интерфейс данные всегда передаются в двоичном коде. Вопрос как эти данные интерпретировать, как воспринимать. Например, передан двоичный код "01000001" (десятичный 65). Как его отобразить на экране? Может быть передано число 65 и на экране надо вывести "65". А может это код буквы "A", тогда на экране надо написать "A". Просто необходимо знать в каком формате передаются данные.

В классе **Serial данные могут передаваться в двух форматах:**

- как бинарный код;
- как **ASCII** символы.

ASCII

кодировка

ASCII — это таблица кодировки символов, в которой каждой букве, числу или знаку соответствует определенное число. В стандартной таблице ASCII 128 символов, пронумерованных от 0 до 127. В них входят латинские буквы, цифры, знаки препинания и управляющие символы.

ASCII Table

| Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char |
|-----|-----|-----|------|-----|-----|-----|---------|-----|-----|-----|------|-----|-----|-----|------|
| 0 | 0 | 0 | | 32 | 20 | 40 | [space] | 64 | 40 | 100 | @ | 96 | 60 | 140 | ` |
| 1 | 1 | 1 | | 33 | 21 | 41 | ! | 65 | 41 | 101 | A | 97 | 61 | 141 | a |
| 2 | 2 | 2 | | 34 | 22 | 42 | " | 66 | 42 | 102 | B | 98 | 62 | 142 | b |
| 3 | 3 | 3 | | 35 | 23 | 43 | # | 67 | 43 | 103 | C | 99 | 63 | 143 | c |
| 4 | 4 | 4 | | 36 | 24 | 44 | \$ | 68 | 44 | 104 | D | 100 | 64 | 144 | d |
| 5 | 5 | 5 | | 37 | 25 | 45 | % | 69 | 45 | 105 | E | 101 | 65 | 145 | e |
| 6 | 6 | 6 | | 38 | 26 | 46 | & | 70 | 46 | 106 | F | 102 | 66 | 146 | f |
| 7 | 7 | 7 | | 39 | 27 | 47 | ' | 71 | 47 | 107 | G | 103 | 67 | 147 | g |
| 8 | 8 | 10 | | 40 | 28 | 50 | (| 72 | 48 | 110 | H | 104 | 68 | 150 | h |
| 9 | 9 | 11 | | 41 | 29 | 51 |) | 73 | 49 | 111 | I | 105 | 69 | 151 | i |
| 10 | A | 12 | | 42 | 2A | 52 | * | 74 | 4A | 112 | J | 106 | 6A | 152 | j |
| 11 | B | 13 | | 43 | 2B | 53 | + | 75 | 4B | 113 | K | 107 | 6B | 153 | k |
| 12 | C | 14 | | 44 | 2C | 54 | , | 76 | 4C | 114 | L | 108 | 6C | 154 | l |
| 13 | D | 15 | | 45 | 2D | 55 | - | 77 | 4D | 115 | M | 109 | 6D | 155 | m |
| 14 | E | 16 | | 46 | 2E | 56 | . | 78 | 4E | 116 | N | 110 | 6E | 156 | n |
| 15 | F | 17 | | 47 | 2F | 57 | / | 79 | 4F | 117 | O | 111 | 6F | 157 | o |
| 16 | 10 | 20 | | 48 | 30 | 60 | 0 | 80 | 50 | 120 | P | 112 | 70 | 160 | p |
| 17 | 11 | 21 | | 49 | 31 | 61 | 1 | 81 | 51 | 121 | Q | 113 | 71 | 161 | q |
| 18 | 12 | 22 | | 50 | 32 | 62 | 2 | 82 | 52 | 122 | R | 114 | 72 | 162 | r |
| 19 | 13 | 23 | | 51 | 33 | 63 | 3 | 83 | 53 | 123 | S | 115 | 73 | 163 | s |
| 20 | 14 | 24 | | 52 | 34 | 64 | 4 | 84 | 54 | 124 | T | 116 | 74 | 164 | t |
| 21 | 15 | 25 | | 53 | 35 | 65 | 5 | 85 | 55 | 125 | U | 117 | 75 | 165 | u |
| 22 | 16 | 26 | | 54 | 36 | 66 | 6 | 86 | 56 | 126 | V | 118 | 76 | 166 | v |
| 23 | 17 | 27 | | 55 | 37 | 67 | 7 | 87 | 57 | 127 | W | 119 | 77 | 167 | w |
| 24 | 18 | 30 | | 56 | 38 | 70 | 8 | 88 | 58 | 130 | X | 120 | 78 | 170 | x |
| 25 | 19 | 31 | | 57 | 39 | 71 | 9 | 89 | 59 | 131 | Y | 121 | 79 | 171 | y |
| 26 | 1A | 32 | | 58 | 3A | 72 | : | 90 | 5A | 132 | Z | 122 | 7A | 172 | z |
| 27 | 1B | 33 | | 59 | 3B | 73 | ; | 91 | 5B | 133 | [| 123 | 7B | 173 | { |
| 28 | 1C | 34 | | 60 | 3C | 74 | < | 92 | 5C | 134 | \ | 124 | 7C | 174 | |
| 29 | 1D | 35 | | 61 | 3D | 75 | = | 93 | 5D | 135 |] | 125 | 7D | 175 | } |
| 30 | 1E | 36 | | 62 | 3E | 76 | > | 94 | 5E | 136 | ^ | 126 | 7E | 176 | ~ |
| 31 | 1F | 37 | | 63 | 3F | 77 | ? | 95 | 5F | 137 | _ | 127 | 7F | 177 | |

Основные функции класса

Serial.

Serial.begin

in

Разрешает работу порта **UART** и задает скорость обмена в бод (бит в сек). Для задания скорости передачи данных рекомендуется использовать стандартные значения из таблицы.

```
Serial.begin(38400); // инициализация порта, скорость 38400 бод
```

Serial.end

Отключает порт **UART**, освобождает выводы **RX** и **TX**.

```
Serial.end(); // закрыть порт UART
```

Serial. Available

Возвращает количество байт, принятых последовательным портом и записанных в буфер. Буфер последовательного порта может хранить до 64 байт. В случае пустого буфера возвращает 0.

```
int n;  
n = Serial.available(); // в n число принятых байтов
```

Serial.print

Выводит данные через последовательный порт UART в виде ASCII символов. Функция имеет различные формы вызова для разных форматов и типов данных.

```
Serial.print("Буквы"); // выводит строку "Буквы"
```


| | |
|----------------------|---|
| print(char d) | <p>Если аргумент типа char выводит в порт код символа</p> <pre><i>char d= 83;</i> <i>Serial.print(d); // выводит код 83 (символ S)</i> <i>Serial.print('S'); // выводит код 83 (символ S)</i></pre> |
| print(byte d) | <p>Данные типа byte выводятся кодом числа</p> <pre><i>byte d= 83;</i> <i>Serial.print(d); // выводит код 83 (символ S)</i> <i>Serial.print(byte(83)); // выводит код 83 (символ S)</i></pre> |
| print(int d) | <p>Если аргумент – целый тип, то выводит строку с десятичным представлением числа</p> <pre><i>int d= 83;</i> <i>Serial.print(d); // выводит строку "83"</i> <i>Serial.print(83); // выводит строку "83"</i></pre> |
| print(float) | <p>Вещественные типы выводятся символами ASCII, два знака после запятой</p> <pre><i>float d= 7.65432;</i> <i>Serial.print(d); // выводит строку "7.65"</i> <i>Serial.print(7.65432); // выводит строку "7.65"</i></pre> |

| | |
|---------------------------|--|
| print(int d, DEC) | <p>Выводит строку ASCII - десятичное представление числа</p> <pre><i>int d= 83;</i> <i>Serial.print(d, DEC); // вывод строки "83"</i></pre> |
| print(int d, HEX) | <p>Выводит строку ASCII – шестнадцатеричное представление числа</p> <pre><i>int d= 83;</i> <i>Serial.print(d, HEX); // вывод строки "53"</i></pre> |
| print(int d, OCT) | <p>Выводит строку ASCII – восьмеричное представление числа</p> <pre><i>int d= 83;</i> <i>Serial.print(d, OCT); // вывод строки "123"</i></pre> |
| print(int d, BIN) | <p>Выводит строку ASCII – двоичное представление числа</p> <pre><i>int d= 83;</i> <i>Serial.print(d, BIN); // вывод строки "01010011"</i></pre> |
| print(int d, BYTE) | <p>Выводит код младшего байта числа</p> <pre><i>int d= 0x0283;</i> <i>Serial.print(d, BYTE); // вывод числа 83 (код символа S)</i></pre> |
| print(float d, N) | <p>Для вещественных чисел параметр N задает количество цифр после запятой.</p> <pre><i>Serial.print(7.65432, 0); // выводит строку "7"</i> <i>Serial.print(7.65432, 2); // выводит строку "7.65"</i> <i>Serial.print(7.65432, 4); // выводит строку "7.6543"</i></pre> |

Serial.println

Выводит данные через последовательный порт **UART** в виде **ASCII** символов с добавлением символов переноса строки (**\r**, код **13**) и (**\n**, код **10**). Т.е. следующее сообщение будет отображаться с новой строки. В остальном аналогична функции `print()`.

```
int d= 83;  
Serial.print(d, DEC); // вывод строки "83"  
Serial.println(d, DEC); // вывод строки "83 \r \n"
```

Serial.write

Выводит двоичные данные через последовательный порт **UART**. Возвращает количество переданных байтов

| | |
|-----------------------------|---|
| <code>int write(val)</code> | Передает байт <i>Serial.write(83); // передает байт 83</i> |
| <code>int write(str)</code> | Передает строку, как последовательность байтов <i>int bytesNumber; // число байтов bytesNumber= Serial.write("Строка"); // передает строку "Строка", возвращает длину строки</i> |

Применение класса **Serial.**

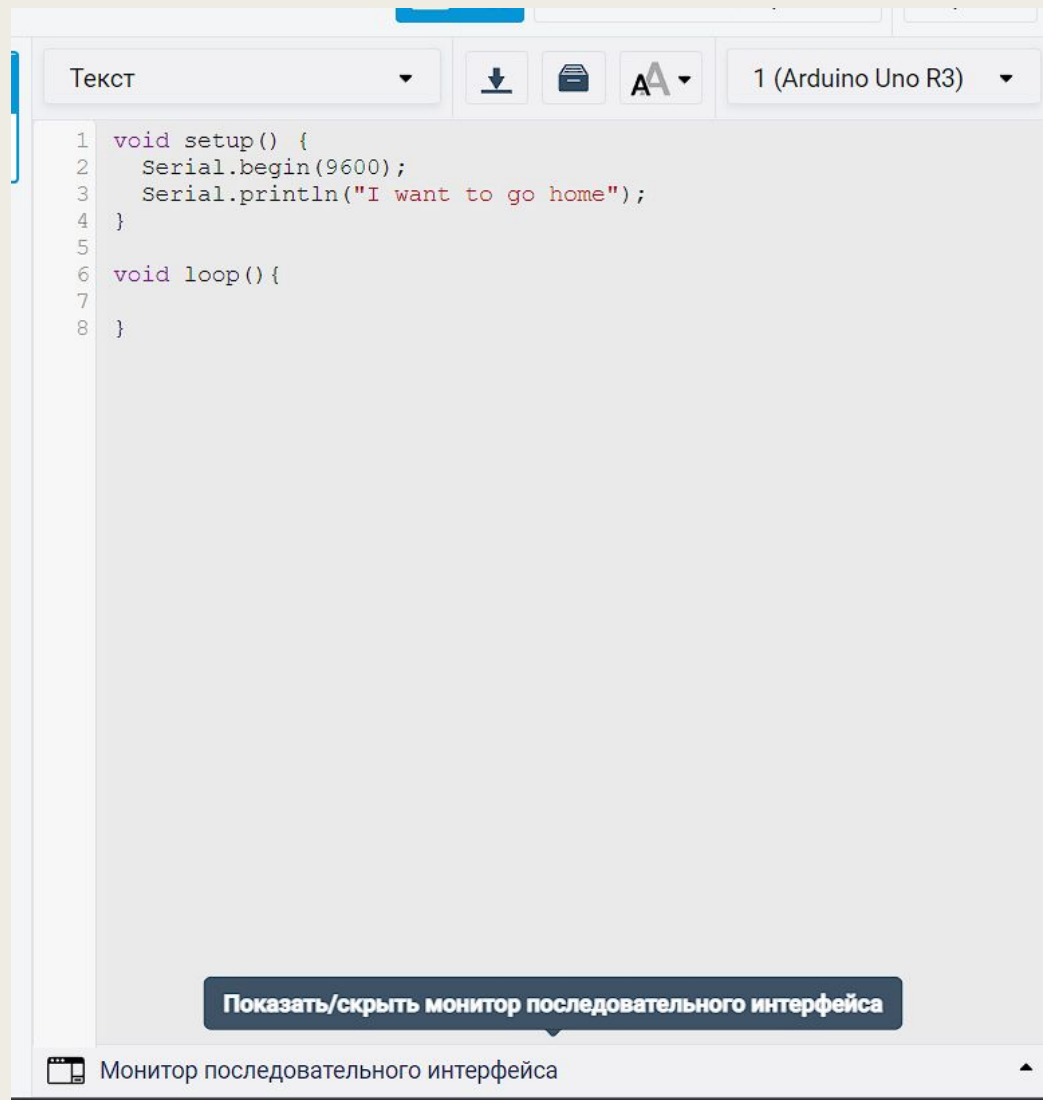
Класс Serial встроенный. Для него не надо искать библиотеку и подключать ее. Чтобы использовать UART достаточно в setup() разрешить работу порта и задать скорость:

```
void setup() {  
  Serial.begin(9600); // инициализируем порт, скорость 9600  
}
```

Теперь можно передавать данные с помощью функций print() или write().

```
Serial.println("Message to monitor"); // сообщение в монитор последовательного порта
```

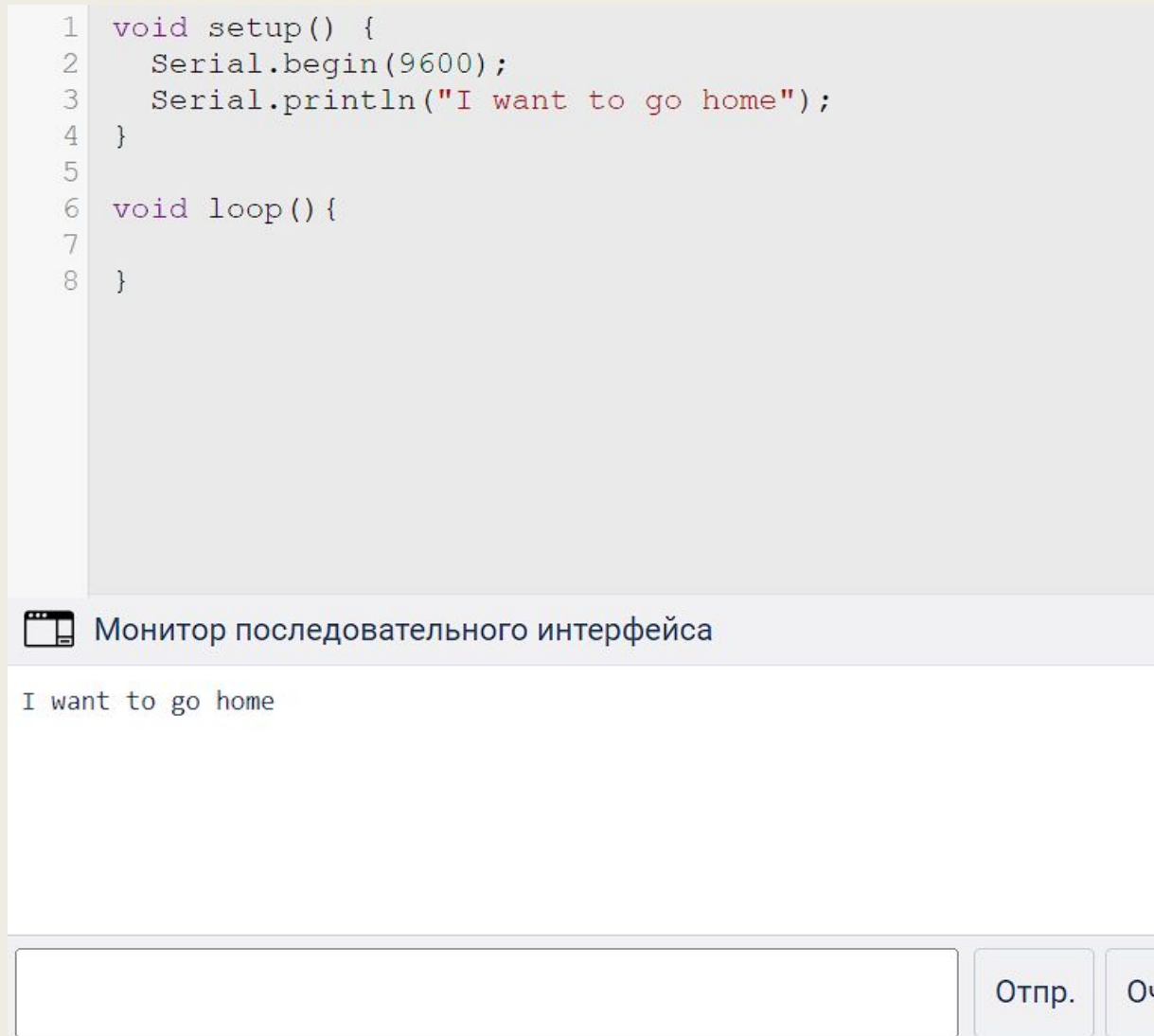
Давайте выведем какое-нибудь сообщение. Это можно сделать в методе `setup()`, так как нам не нужно повторять одну и ту же фразу бесконечно. Метод `loop()` оставляем пустым.



The screenshot shows the Arduino IDE code editor. At the top, there is a toolbar with a dropdown menu set to "Текст", a download icon, a printer icon, a font size selector, and a board selector set to "1 (Arduino Uno R3)". The code editor contains the following code:

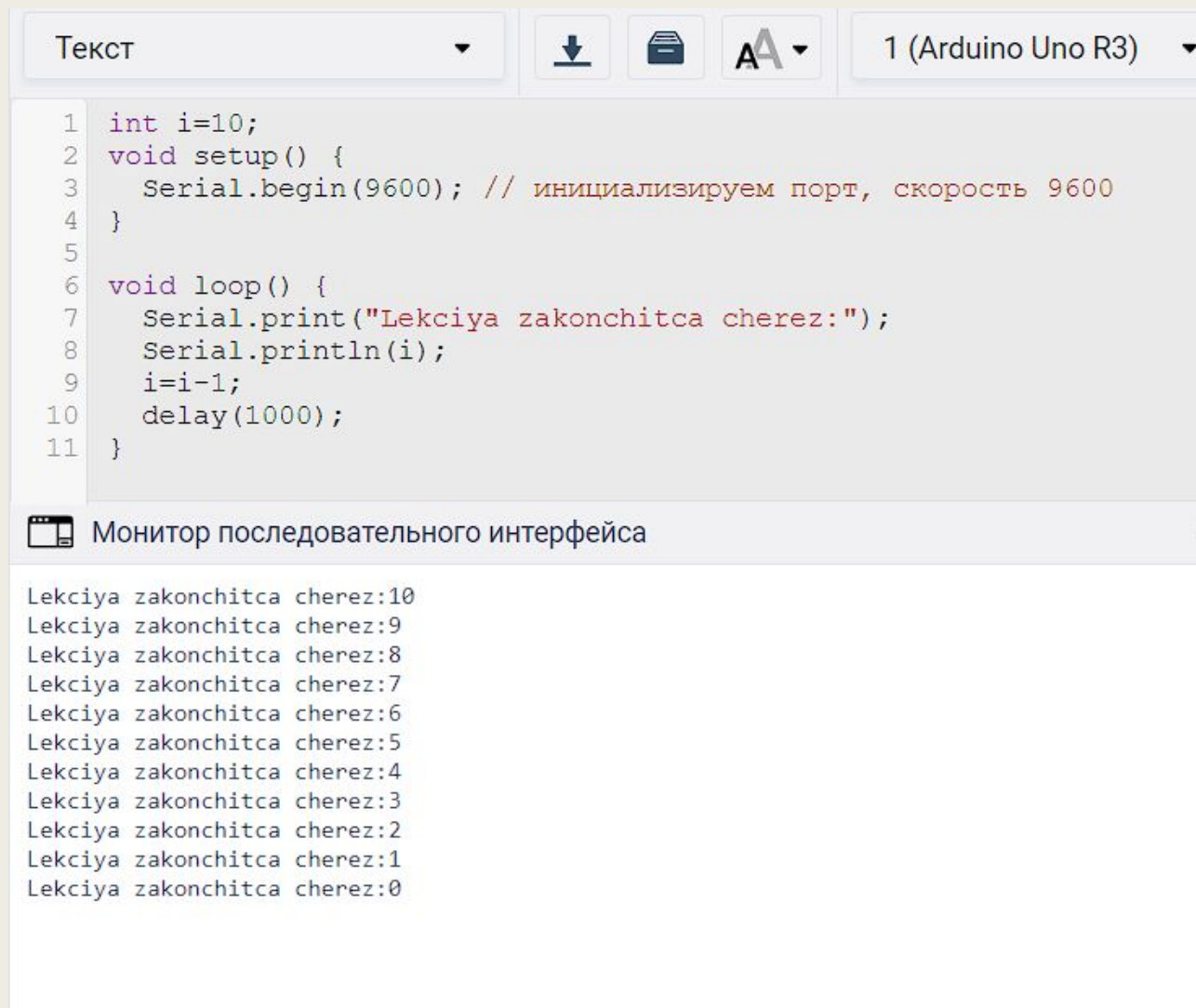
```
1 void setup() {  
2   Serial.begin(9600);  
3   Serial.println("I want to go home");  
4 }  
5  
6 void loop() {  
7  
8 }
```

At the bottom of the editor, there is a button labeled "Показать/скрыть монитор последовательного интерфейса". Below the editor, a status bar shows a monitor icon and the text "Монитор последовательного интерфейса".



The screenshot shows the serial monitor window in the Arduino IDE. The title bar reads "Монитор последовательного интерфейса". The main area of the monitor displays the text "I want to go home". At the bottom, there is an input field and two buttons: "Отпр." and "Ос".

Вот программа, которая каждую секунду выводит в монитор последовательного порта сообщение и переменную.



The image shows a screenshot of the Arduino IDE interface. At the top, there is a toolbar with a dropdown menu set to 'Текст', a download icon, a save icon, a font size icon, and a dropdown menu set to '1 (Arduino Uno R3)'. Below the toolbar is a code editor with the following C++ code:

```
1 int i=10;
2 void setup() {
3   Serial.begin(9600); // инициализируем порт, скорость 9600
4 }
5
6 void loop() {
7   Serial.print("Lekciya zakonchitca cherez:");
8   Serial.println(i);
9   i=i-1;
10  delay(1000);
11 }
```

Below the code editor is a serial monitor window titled 'Монитор последовательного интерфейса'. It displays the output of the program, which consists of ten lines of text, each showing the current value of the variable 'i' and the message 'Lekciya zakonchitca cherez:':

```
Lekciya zakonchitca cherez:10
Lekciya zakonchitca cherez:9
Lekciya zakonchitca cherez:8
Lekciya zakonchitca cherez:7
Lekciya zakonchitca cherez:6
Lekciya zakonchitca cherez:5
Lekciya zakonchitca cherez:4
Lekciya zakonchitca cherez:3
Lekciya zakonchitca cherez:2
Lekciya zakonchitca cherez:1
Lekciya zakonchitca cherez:0
```