1. **Цель работы:**

Работа направлена на понимание особенностей различных типов данных, таких как целые числа, числа с плавающей точкой, символы и булевые значения, а также на отработку навыков отладки программ и мониторинга данных через последовательный порт.

1. **Задание на лабораторную работу:**
2. Исследование типов данных в Arduino IDE. (эти знания были получены из [1])
3. Запустить программу Arduino IDE.
4. Объявить данные в Arduino IDE.
5. Задать разрешённые величины данных в Arduino IDE.
6. Запустить отладку программы в Arduino IDE.
7. Задать неразрешенные величины данных в Arduino IDE.
8. Запустить отладку программы в Arduino IDE.
9. Опять загрузить разрешённые величины данных в Arduino IDE.
10. Вывести значения разрешённых данных на монитор. (эти знания были получены из [4])
11. **Выполнение лабораторной работы:**

Файл sketch.ino:

void setup() {

Serial.begin(9600); // Открываем монитор порта

// Разрешённые величины

double myDouble = 3.1415926535;

int myInt = 32000;

unsigned int myUInt = 65000;

long myLong = 2000000000;

float myFloat = 3.14159;

char myChar = 'A';

byte myByte = 255;

boolean myBool = true;

Serial.println("Разрешённые значения:");

Serial.print("double: "); Serial.println(myDouble, 10);

Serial.print("int: "); Serial.println(myInt);

Serial.print("unsigned int: "); Serial.println(myUInt);

Serial.print("long: "); Serial.println(myLong);

Serial.print("float: "); Serial.println(myFloat, 5);

Serial.print("char: "); Serial.println(myChar);

Serial.print("byte: "); Serial.println(myByte);

Serial.print("boolean: "); Serial.println(myBool);

Serial.println("\nРазмер типов данных в байтах:");

Serial.print("double: "); Serial.println(sizeof(myDouble));

Serial.print("int: "); Serial.println(sizeof(myInt));

Serial.print("unsigned int: "); Serial.println(sizeof(myUInt));

Serial.print("long: "); Serial.println(sizeof(myLong));

Serial.print("float: "); Serial.println(sizeof(myFloat));

Serial.print("char: "); Serial.println(sizeof(myChar));

Serial.print("byte: "); Serial.println(sizeof(myByte));

Serial.print("boolean: "); Serial.println(sizeof(myBool));

delay(5000); // Задержка 5 секунд перед тестированием ошибок

}

void testInvalidValues() {

Serial.println("\nПроверяем недопустимые значения:");

int myInt = 40000; // Превышает 32,767

unsigned int myUInt = -1; // Нельзя отрицательное число

long myLong = 3000000000; // Превышает max long

float myFloat = 1e40; // Слишком большое число

char myChar = 'AB'; // Ошибка: char принимает только 1 символ

byte myByte = 300; // Больше 255

boolean myBool = 5; // Должно быть true/false, но преобразуется

Serial.print("int (недопустимое): "); Serial.println(myInt);

Serial.print("unsigned int (недопустимое): "); Serial.println(myUInt);

Serial.print("long (недопустимое): "); Serial.println(myLong);

Serial.print("float (недопустимое): "); Serial.println(myFloat);

Serial.print("char (недопустимое): "); Serial.println(myChar);

Serial.print("byte (недопустимое): "); Serial.println(myByte);

Serial.print("boolean (недопустимое): "); Serial.println(myBool);

}

void loop() {

delay(10000); // Ждём 10 секунд перед тестом ошибок

testInvalidValues();

while (true); // Останавливаем программу

}

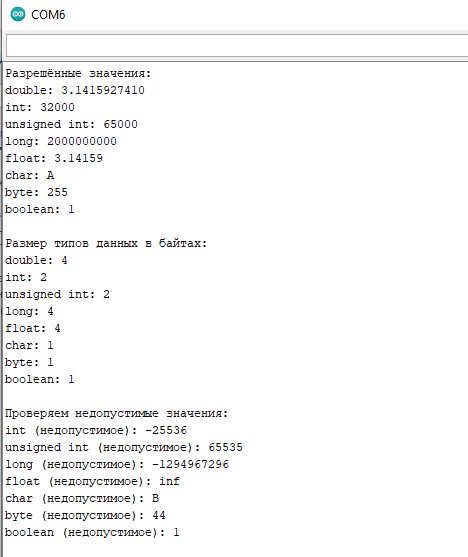
В программе мы проверяли основные типы данных Arduino Nano и их основные значения.

Сначала в функции setup мы задали допустимые значения, а потом в функции testInvalidValues те, что выходят за границы.

Программа не даёт задать неверные значения, поэтому приводит их к допустимому виду. Опишем эти преобразования:

* int (40000) переполнится и станет -25536
* unsigned int (-1) переполнится и станет 65535
* long (3000000000) превратится в -1294967296
* float (1e40) станет бесконечностью
* char ('AB') вызовет ошибку компиляции, и приравняет значение к «В».
* byte (300) станет 44 (переполнение)
* boolean (5) будет считаться как true

Результат запуска программы в мониторе порта:



1. **Выводы:**

В ходе исследования было продемонстрировано правильное использование различных типов данных в Arduino IDE, проверены их допустимые диапазоны значений и выявлены ситуации выхода за эти границы (эти знания были получены из [1]). Работа позволила закрепить знания о типах данных и их применении в микроконтроллерах (эти знания были получены из [2] и [3]), а также развить навыки диагностики и устранения ошибок в программах для Arduino (эти знания были получены из [4]).

1. **Используемые источники:**
2. Петин В.А.: Проекты с использованием контроллера. - 464 стр. - Санкт-Петербург - 2015 г.
3. Аверин В.Н.: Практическая энциклопедия Arduino. — 352 стр. — Москва — 2017 г.
4. Башар А.М.: Основы программирования микроконтроллеров Arduino. — 256 стр. — Ростов-на-Дону — 2018 г.

4. <https://www.arduino.cc>