

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Иркутский национальный исследовательский технический
университет»
Институт информационных технологий и анализа данных

О Т Ч Ъ Т

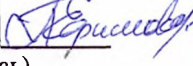
о прохождении учебной практики
(вид практики: учебная/производственная)
технологической (проектно-технологической) практики
(тип практики: технологическая/научно-исследовательская работа/преддипломная и др.)
в ИРНИТУ
(наименование профильной организации)





Резюме на superjob.ru



Резюме на hh.ru

Обучающегося Ефимова П.О., ИСИБ-24-1 
(ФИО, группа, подпись)

Руководитель практики от института ИТиАД
Кононенко Р.В., доцент института ИТиАД 
(ФИО, должность, подпись)

Руководитель образовательной программы
Кононенко Р.В., доцент института ИТиАД 
(ФИО, должность, подпись)

Оценка по практике зачет
Кононенко Р.В. Тко 30.09.2025
(ФИО, подпись, дата)

Содержание отчета на 37 стр.
Приложение к отчету на 0 стр

Содержание

Введение	3
Задание №1.....	4
Задание №2.....	7
Задание №3.....	9
Задание №4.....	12
Задание №5.....	15
Задание №6.....	18
Задание №7.....	19
Задание №8.....	20
Задание №9.....	23
Задание №10.....	27
Задание №11.....	31
Отзыв о посещении филиала АО «СО ЕЭС» Иркутское РДУ	34
Отзыв о посещении компании ISPsystem	35
Заключение	36
Список литературы.....	37

Введение

Учебная практика на базе Федерального государственного бюджетного образовательного учреждения высшего образования «Иркутский национальный исследовательский технический университет» проходила в период с 16 по 28 июня 2025 года. Основная цель практики заключалась в закреплении знаний, полученных в ходе изучения профильных дисциплин, а также в развитии умений применять их на практике.

Особое внимание уделялось программированию на языке C++, формированию навыков анализа и решения задач, а также самостоятельному изучению нового материала. Для этого было выполнено одиннадцать заданий, каждое из которых направлено на развитие конкретных тем: от работы с алгоритмами и структурами данных до освоения инструментов, используемых в области информационных технологий.

Прохождение практики способствовало применению теоретических умений на практике, а также углублению профессиональных навыков

Задание №1

Незнайка в своей экспедиции на Луну оказался на вершине лунной горы. Спуск вниз опасен, поэтому он взял с собой карту склона горы, где числами обозначено, сколько минут требуется на этот участок маршрута. Спуск происходит сверху вниз на один из соседних участков. Пример наиболее короткого маршрута выделен красным цветом, сумма чисел = 10. Напишите программу, рассчитывающую минимальное время спуска (сумму чисел в пути с вершины до основания).

Алгоритм программы:

- 1) Ввод данных: программа считывает высоту пирамиды и генерирует её заполненную числами;
- 2) Расчёт минимальной суммы пути: создаётся копия пирамиды, по которой программа проходит снизу вверх и для каждого уровня обновляет минимальную сумму пути;
- 3) Восстановление кратчайшего пути: программа проходит с вершины пирамиды до основания, выбирая минимальную сумму на уровне из записанных и добавляет значение в путь;
- 4) Восстановление пути: начиная с вершины и на каждом уровне идем в направлении меньшей суммы.
- 5) Вывод: значения минимального пути

Код программы:

```
1  #include <iostream>
2  #include <vector>
3
4  int main() {
5      int n;
6      std::cout << "Введите число: ";
7      std::cin >> n;
8
9      std::vector<std::vector<int>> pyr(n);
10     for (int i = 0; i < n; i++) {
11         for (int j = 0; j <= i; j++) {
12             pyr[i].push_back(std::rand() % 1000);
13         }
14     }
15
16     std::cout << "Пирамида: " << '\n';
17     for (int i = 0; i < n; i++) {
18         for (int j = 0; j <= i; j++) {
19             std::cout << pyr[i][j] << " ";
20         }
21     }
22     std::cout << '\n';
23
24     std::vector<std::vector<int>> sum = pyr;
25     for (int i = n - 2; i >= 0; i--) {
26         for (int j = 0; j < i; j++) {
27             sum[i][j] += std::min(sum[i+1][j], sum[i+1][j+1]);
28         }
29     }
30
31     std::cout << "Минимальная сумма: " << sum[0][0] << '\n';
32
33 }
```

Рисунок 1 - 1 часть кода задачи №1

```

34         std::cout << pyr[0][0] << " ";
35     int j = 0;
36     for (int i = 1; i < n; i++) {
37         if (sum[i][j] < sum[i][j+1]) {
38             std::cout << pyr[i][j] << " ";
39         } else {
40             std::cout << pyr[i][j+1] << " ";
41             j++;
42         }
43     }
44 }
45 return 0;
46 }

```

Рисунок 2 - 2 часть кода задачи №1

Таблица тестов:

Таблица 1 – Тесты для 1 задания

Ввод	6	3
Вывод	Пирамида: 383 886 777 915 793 335 386 492 649 421 362 27 690 59 763 926 540 426 172 736 211 Минимальная сумма: 383 383 777 335 421 59 172	Пирамида: 383 886 777 915 793 335 Минимальная сумма: 383 383 777 335

Ввод образца

5
356
123 342
652 712 145
152 354 756 827
87 930 762 76 382

Ваш Результат

Введите число: Пирамида:
383
886 777
915 793 335
386 492 649 421
362 27 690 59 763
Минимальная сумма: 383
383 777 335 421 59

Рисунок 3 - Пример теста из задачи № 1

Задание №2

После метеоритной атаки компьютерная сеть для управления лунными заводами разбилась на части, нужно объединить её в единое целое. Каждый фрагмент сети представлен в виде ненаправленного графа.

Вам известно общее число вершин графа (узлы сети, не более 1000) и набор рёбер (сохранившиеся линии связи, не более 1000).

Определите, какое минимальное число линий связи нужно дополнительно построить, чтобы сеть стала единой.

Алгоритм программы:

- 1) Ввод данных: программа считывает количество вершин и количество ребер и создаёт массив в формате системы непересекающихся множеств;
- 2) Обработка рёбер: для каждого ребра читается пара вершин, которая объединяется в множество вершин;
- 3) Подсчёт компонент связности: проходим по всем вершинам, начиная с 1, считаем количество корневых компонент, представляющих отдельные компоненты связности;
- 4) Вывод результата: число компонент меньше на 1 является минимальным количеством рёбер для соединения всех компонент

Код программы:

```
1  #include <iostream>
2  #include <vector>
3  #include <numeric>
4
5  struct DSU {
6      std::vector<int> parent;
7      DSU(int n) {
8          parent.resize(n + 1);
9          std::iota(parent.begin(), parent.end(), 0);
10     }
11
12     int find(int i) {
13         if (parent[i] == i)
14             return i;
15         return parent[i] = find(parent[i]);
16     }
17
18     void unite(int i, int j) {
19         int root_i = find(i);
20         int root_j = find(j);
21         if (root_i != root_j)
22             parent[root_i] = root_j;
23     }
24 };
25
26 int main() {
27     std::ios_base::sync_with_stdio(false);
28     std::cin.tie(NULL);
29     int n, m;
30     std::cin >> n >> m;
31 }
```

Рисунок 4 - 1 часть кода задачи № 2

```

32     DSU dsu(n);
33
34     for (int k = 0; k < m; ++k) {
35         int u, v;
36         std::cin >> u >> v;
37         dsu.unite(u, v);
38     }
39
40     int components = 0;
41     for (int i = 1; i <= n; ++i) {
42         if (dsu.parent[i] == i)
43             components++;
44     }
45
46     std::cout << components - 1 << std::endl;
47
48     return 0;
49 }

```

Рисунок 5 - 2 часть кода задачи № 2

Таблица тестов:

Таблица 2 – Тесты для 2 задания

Ввод	15 5 1 7 8 2 9 1 4 3 3 6	7 4 1 2 2 3 4 5 6 7
Вывод	9	2

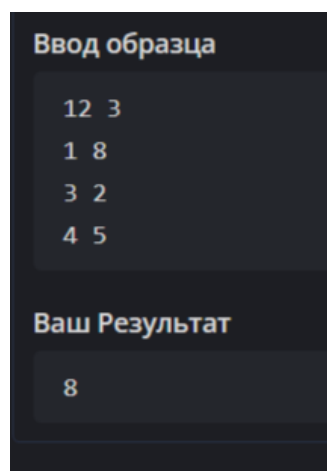


Рисунок 6 - Пример теста из задачи №2

Задание №3

В Иркутске раз в году наступает зима. Несмотря на то, что событие это довольно регулярное, оно всегда внезапно. Снег буквально заваливает все улицы, не давая проехать на чём-то меньше трактора. В этом году терпение лопнуло и специальным указом был создан кризисный центр по борьбе с сугробами. Центру были переданы спутники, лазеры, метеорологические зонды и несколько десятков лопат.

Вам поручено возглавить отдел разведки снежной ситуации и быть способным чрезвычайно быстро отвечать на запросы центра. Сам город состоит из нескольких, расположенных подряд, улиц, каждая из которых абсолютна похожа на любую другую.

1) Информация о снеге передается вам в виде тройки чисел – 1 в качестве идентификатора события, уникального индекса улицы и количество миллиметров выпавшего снега.

2) Запросы в свою очередь так же имеют вид тройки чисел – 2 в качестве идентификатора события, индекс улицы с которой нужно суммировать количество выпавшего снега и индекс улицы, по которую нужно суммировать, крайние улицы должны быть включены.

Алгоритм программы:

- 1) Ввод данных: программа считывает размер массива и количество операций.
- 2) Обработка запросов: принимаются запросы двух типов:
 - Тип 1: обновляет количество снега на конкретной улице. Значение записывается в соответствующий индекс массива.
 - Тип 2: вычисляет и выводит сумму снега на отрезке улиц. Программа проходит по элементам массива в этом диапазоне и суммирует значения.
- 3) Вывод результатов: результат суммирования по запросам типа 2.

Код программы:

```

1  #include <iostream>
2  #include <vector>
3
4  class FenwickTree {
5  private:
6      std::vector<long long> bit;
7      int size;
8
9  public:
10     FenwickTree(int n) : size(n), bit(n + 1, 0) {}
11
12     void update(int idx, int delta) {
13         for (; idx <= size; idx += idx & -idx) {
14             bit[idx] += delta;
15         }
16     }
17
18     long long query(int idx) {
19         long long sum = 0;
20         for (; idx > 0; idx -= idx & -idx) {
21             sum += bit[idx];
22         }
23         return sum;
24     }
25 };
26
27 int main() {
28     std::ios_base::sync_with_stdio(false);
29     std::cin.tie(NULL);
30
31     int n, k;
32     std::cin >> n >> k;
33
34     FenwickTree ft(n);
35     std::vector<long long> results;
36

```

Рисунок 7 - 1 часть кода задачи № 3

```

36
37     for (int op = 0; op < k; ++op) {
38         int type;
39         std::cin >> type;
40         if (type == 1) {
41             int i, x;
42             std::cin >> i >> x;
43             ft.update(i, x);
44         } else {
45             int u, r;
46             std::cin >> u >> r;
47             results.push_back(ft.query(r) - ft.query(u - 1));
48         }
49     }
50
51     for (long long res : results) {
52         std::cout << res << std::endl;
53     }
54
55     return 0;
56 }

```

Рисунок 8 - 2 часть кода задачи № 3

Таблица тестов:

Таблица 3 – Тесты для 3 задания

Ввод	6 5 2 1 6 1 3 2 2 2 4 1 6 3 2 1 6	5 3 1 3 7 1 1 4 2 1 5
Вывод	0 2 5	11

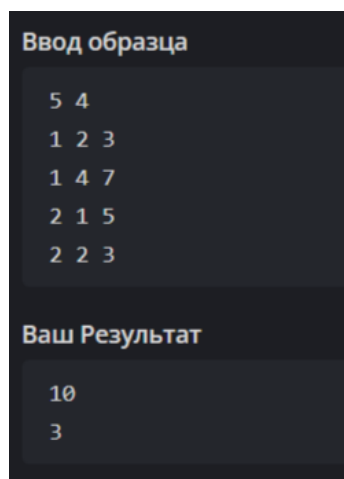


Рисунок 9 - Пример теста из задачи №3

Задание №4

Перестановка P длины n — это упорядоченный набор, содержащий числа от 1 до n , каждое из которых входит в него ровно один раз. Например, перестановкой длины 13 является набор (5 11 13 12 6 1 8 4 10 9 7 2 3). Само название говорит о том, для чего предназначен этот объект. Например, можно при помощи перестановки букв зашифровать слово. Для примера возьмем приведенную выше перестановку и слово *transposition*, которое состоит тоже из 13 букв. Далее, следуя перестановке, на первую позицию поставим пятую букву слова, на вторую – одиннадцатую букву и так далее. В итоге получим *sinoptsnrtiora*. К этому слову снова применим эту же перестановку и получим *roartsnoitsin*. Повторив эти стадии шифрования k раз, получим зашифрованное сообщение.

Вам дано зашифрованное таким образом слово, шифрующая перестановка P и число k . Необходимо восстановить слово.

Алгоритм программы:

- 1) Ввод данных: считывается два числа: размер перестановки (n) и количество применений обратной перестановки (k), затем считываются перестановка P размером n и зашифрованная строка;
- 2) Обратная перестановка: для каждого элемента $P[i]$ записываем в P_inv значение i на позицию $P[i]-1$, получая перестановку обратную к исходной. Теперь в $result[i]$ находится элемент $P[i]$;
- 3) Процесс дешифровки строки:
 - Строится обратная перестановка P_inv ;
 - Применяется обратная перестановка k раз к зашифрованной строке;
- 4) Результат: программа выводит восстановленное слово, которое является результатом декодирования зашифрованного слова.

Код программы:

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4
5  using namespace std;
6
7  vector<int> inverse_permutation(const vector<int>& P) {
8      int n = P.size();
9      vector<int> P_inv(n);
10     for (int i = 0; i < n; ++i) {
11         P_inv[P[i] - 1] = i;
12     }
13     return P_inv;
14 }
15 string apply_permutation(const string& s, const vector<int>& P) {
16     int n = s.size();
17     string result(n, ' ');
18     for (int i = 0; i < n; ++i) {
19         result[i] = s[P[i]];
20     }
21     return result;
22 }
23 string decrypt(const string& encrypted, const vector<int>& P, int k) {
24     int n = encrypted.size();
25     vector<int> P_inv = inverse_permutation(P);
26     string current = encrypted;
27     for (int i = 0; i < k; ++i) {
28         current = apply_permutation(current, P_inv);
29     }
30     return current;
31 }
32

```

Рисунок 10 - 1 часть кода задачи № 4

```

32
33 int main() {
34     int n, k;
35     cin >> n >> k;
36     vector<int> P(n);
37     for (int i = 0; i < n; ++i) {
38         cin >> P[i];
39     }
40     string encrypted;
41     cin.ignore();
42     getline(cin, encrypted);
43     string original = decrypt(encrypted, P, k);
44     cout << original << endl;
45     return 0;
46 }

```

Рисунок 11 - 2 часть кода задачи № 4

Таблица тестов:

Таблица 4 – Тесты для 4 задания

Ввод	13 2 5 11 13 12 6 1 8 4 10 9 7 2 3 poartsnoitsin	5 1 2 3 4 5 1 brock
Вывод	transposition	kbroc

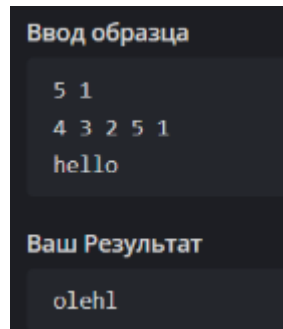


Рисунок 12 - Пример теста из задачи №4

Задание №5

Дана матрица, состоящая из 1 и 0. Значениями 1 в матрице нарисована некоторая фигура. Необходимо определить координаты верхнего левого и нижнего правого углов параллельного осям ограничивающего прямоугольника, т.е. такого прямоугольника, минимального размера, в который фигура помещается полностью и при этом ни одна точка исходной фигуры не попадает на стороны прямоугольника.

Алгоритм программы:

1) Ввод данных:

- Чтение высоты матрицы (h) и ширины матрицы (w);
- Ввод и чтение матрицы размером h x w, состоящую из 0 и 1.

2) Инициализация переменных:

- `int min_row = INT_MAX, min_col = INT_MAX` – минимальные координаты;
- `int max_row = INT_MIN, max_col = INT_MIN` – максимальные координаты.

3) Поиск ограничивающего прямоугольника: проходим по каждому элементу матрицы. Если элемент равен 1, обновляем границы:

- `min_row = min(min_row, i);`
- `max_row = max(max_row, i);`
- `min_col = min(min_col, j);`
- `max_col = max(max_col, j);`

4) Вывод: координаты левого верхнего угла (`min_row, min_col`) и правого верхнего угла (`max_row, max_col`).

Код программы:

```

1  #include <iostream>
2  #include <vector>
3  #include <climits>
4
5  using namespace std;
6
7  ✓ int main() {
8      int h, w;
9      cin >> h >> w;
10
11     vector<vector<int>> matrix(h, vector<int>(w));
12     for (int i = 0; i < h; ++i) {
13         for (int j = 0; j < w; ++j) {
14             cin >> matrix[i][j];
15         }
16     }
17
18     int min_row = INT_MAX, min_col = INT_MAX;
19     int max_row = INT_MIN, max_col = INT_MIN;
20
21     for (int i = 0; i < h; ++i) {
22         for (int j = 0; j < w; ++j) {
23             if (matrix[i][j] == 1) {
24                 if (i < min_row) min_row = i;
25                 if (j < min_col) min_col = j;
26                 if (i > max_row) max_row = i;
27                 if (j > max_col) max_col = j;
28             }
29         }
30     }
31
32     cout << min_row << " " << min_col << " " << max_row << " " << max_col << endl;
33
34     return 0;
35 }

```

Рисунок 13 - Код задачи № 5

Таблица тестов:

Таблица 5 – Тесты для 5 задания

Ввод	5 5 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0	6 6 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0
Вывод	1 2 3 3	1 1 4 3

Задание №6

В школьном кружке робототехники есть два вида микроконтроллеров (условно тип А и тип В) и два вида модулей управления мотором (условно тип 1 и тип 2). Выяснилось, что контроллер типа В и модуль управления типа 2 несовместимы. Использование микроконтроллеров и модулей управления в других комбинациях возможно. Имеется а микроконтроллеров типа А, b микроконтроллеров типа В, x модулей управления типа 1 и y модулей типа 2. Определите, какое максимальное число работающих пар из микроконтроллера и модуля управления мотором можно составить. Ваша программа должна ответить на n запросов.

Алгоритм программы:

- 1) Так как, контроллеры типа В не совместимы с модулями типа 2, то мы можем совместить их только с 1 типом модулей управления.
- 2) Оставшиеся модули типа 1 и все модули типа 2 можно использовать с контроллерами типа А.
- 3) Тогда общее количество пар – это сумма пар котроллер типа В + модуль 1 и контроллер типа А + модуль 1 или 2.

Код программы:

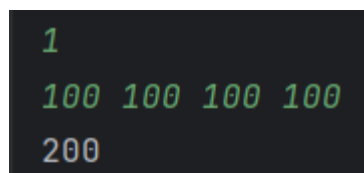
```
1 n=int(input())
2 otvet=[]
3 for i in range (n):
4     a,b,x,y=map(int, input().split())
5     otvet.append(min(b,x)+abs(max(b,x)-(min(b,x)))+min((a - abs(max(b,x)-(min(b,x)))), y))
6 otvet=[str(x) for x in otvet]
7 print(' '.join(otvet))
```

Рисунок 14 - Код задачи № 6

Таблица тестов:

Таблица 6 – Тесты для 6 задания

Ввод	2	2
	7 4 9 3	4 0 2 5
	5 5 6 6	0 5 3 0
Вывод	11 10	4 3



```
1
100 100 100 100
200
```

Рисунок 11 - Пример теста из задачи №6

Задание №7

На компьютере работника автосервиса нашли файл с последовательностью автомобильных номеров, обслуживающихся в этом автосервисе. Так как файл был поврежден, некоторые данные отображаются неверно. Нужно определить, какие из них остались невредимыми.

Автомобильным номером является строка из шести символов. Первый символ – заглавная латинская буква, далее следует 3 цифры, и после – две заглавные латинские буквы. Например, строка "P142EQ" является номером. Вам будет дана строка, состоящая из шести символов, необходимо ответить, является ли строка автомобильным номером.

Алгоритм программы:

Считываем строку и проверяем, соответствует ли она формату буква-цифра-цифра-цифра-буква-буква.

Код программы:

```
1 s=input()
2 a="QWERTYUIOPASDFGHJKLZXCVBNM"
3 b="0123456789"
4 if s[0] not in a: print ('no')
5 else:
6     if s[1] not in b or s[2] not in b or s[3] not in b:print ('no')
7     else:
8         if s[4] not in a or s[5] not in a:
9             print ('no')
10        else: print('yes')
```

Рисунок 16 - Код задачи № 7

Таблица тестов:

Таблица 7 – Тесты для 7 задания

Ввод	K910YR	H64VRE
Вывод	Yes	No

Задание №8

Составить светодиодную матрицу размером не менее 8 на 8 светодиодов. На матрицу вывести инфографику с различными динамично меняющимися изображениями.

Алгоритм программы:

- 1) Инициализация: настраиваем матрицу размером 8x8 (64 светодиода)
- 2) Цикл из анимаций:
 - Анимация 1: синяя точка последовательно проходит все 64 позиции с задержкой в 50 мс между шагами
 - Анимация 2: цвета плавно меняются с задержкой в 50 мс (hue увеличивается на 5 каждый раз)
 - Анимация 3: 10 случайных вспышек с задержкой в 200 мс

Код программы:

```
1  #include <FastLED.h>
2
3  #define LED_PIN
4  #define MATRIX_WIDTH 8
5  #define MATRIX_HEIGHT 8
6  #define NUM_LEDS (MATRIX_WIDTH * MATRIX_HEIGHT)
7
8  CRGB leds[NUM_LEDS];
9  int getLedIndex(int x, int y) {
10     if (y % 2 == 0) {
11         return y * MATRIX_WIDTH + x;
12     } else {
13         return y * MATRIX_WIDTH + (MATRIX_WIDTH - 1 - x);
14     }
15 }
16
17 void setup() {
18     FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS);
19     FastLED.setBrightness(50);
20     Serial.begin(9600);
21 }
22
23 void loop() {
24     runningDot();
25     colorWaves();
26     randomSparkles();
```

Рисунок 17 – Первая часть кода задачи №8

```

27 }
28
29 void runningDot() {
30     for(int i = 0; i < NUM_LEDS; i++) {
31         // Гасим все светодиоды
32         fill_solid(leds, NUM_LEDS, CRGB::Black);
33
34         leds[i] = CRGB::Blue;
35
36         FastLED.show();
37         delay(50);
38     }
39 }
40
41 void colorWaves() {
42     static uint8_t hue = 0;
43     for(int x = 0; x < MATRIX_WIDTH; x++) {
44         for(int y = 0; y < MATRIX_HEIGHT; y++) {
45             int index = getLedImage(x, y);
46             leds[index] = CHSV(hue + (x * 10) + (y * 10), 255, 255);
47         }
48     }
49     hue += 5;
50     FastLED.show();

```

Рисунок 18 – Вторая часть кода задачи №8

```

50     FastLED.show();
51     delay(50);
52 }
53
54 void randomSparkles() {
55     fill_solid(leds, NUM_LEDS, CRGB::Black);
56
57     for(int i = 0; i < 10; i++) {
58         int pos = random(NUM_LEDS);
59         leds[pos] = CHSV(random(255), 255, 255);
60     }
61
62     FastLED.show();
63     delay(200);

```

Рисунок 19 – Третья часть кода задачи №8

Пример работы программы:

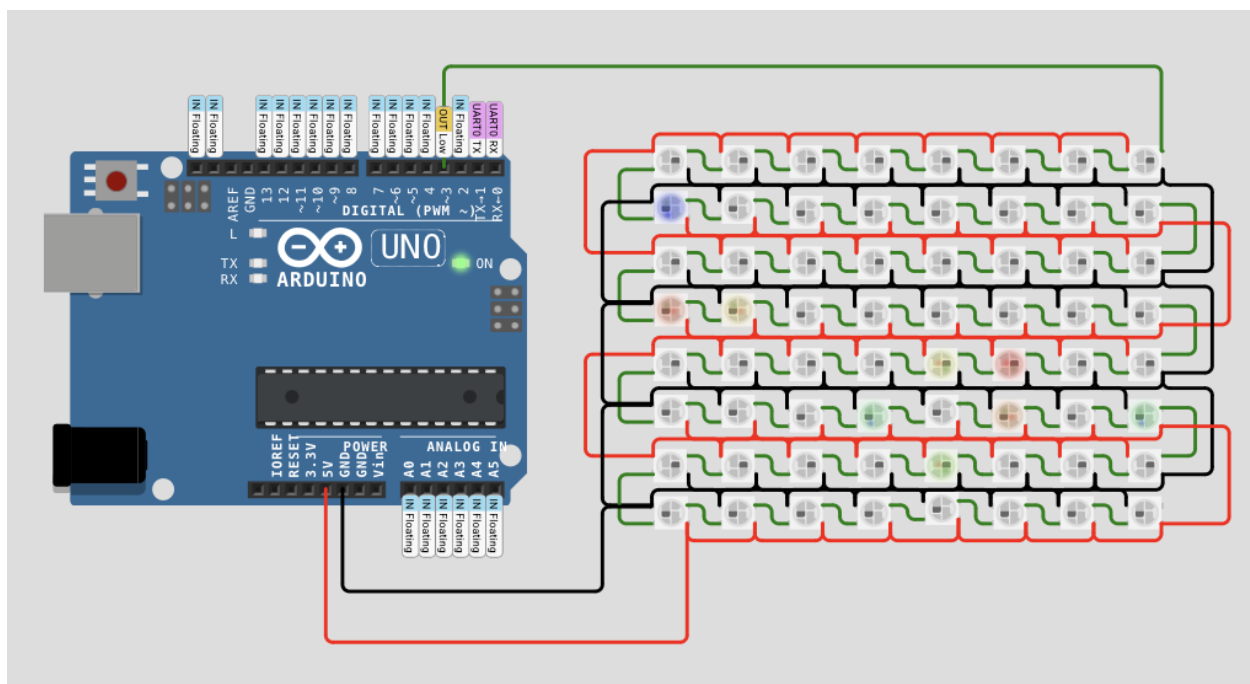


Рисунок 20 - Случайные цвета

Задание №9

Задачи:

- 1) Собрать схему имитирующую работу автоматических дверей
- 2) Подобрать номинал резисторов для светодиодов
- 3) Написать программу для управления процессом работы автоматических дверей.

Схема приведена на рисунке 1.

Зеленый светодиод – двери открываются.

Красный светодиод – двери закрываются.

Фоторезистор имитируют процесс приближения-удаления человека от дверей.

Изменение значений фоторезистора осуществляется при помощи ползунка (рисунок 2), изменение значения фоторезистора доступно только, когда запущен процесс моделирования.

Логика работы программы:

- 1) По умолчанию горит светодиод, имитирующий закрытую дверь
- 2) Микроконтроллер считывает значение фоторезистора с аналогового пина
- 3) Если значение на пине превышает 512, на определенное время загорается светодиод, имитирующий открытую дверь, в последовательный порт выводится сообщение о событии.
- 4) После истечения заданного временного промежутка проверяется значение фоторезистора, если оно всё ещё превышает 512, дверь должна остаться открытой, в противном случае нужно включить индикацию закрытой двери, в последовательный порт выводится сообщение о событии.

Алгоритм программы:

Программа управляет светодиодами для имитации автоматических дверей: если фоторезистор фиксирует низкую освещённость (<512), включается зеленый светодиод (двери открываются); если освещённость высокая (>512), включается красный светодиод (двери закрываются).

Код программы:

```
// Определение пинов для светодиодов и фоторезистора
const int redLedPin = 9;    // Красный светодиод (двери закрыты)
const int greenLedPin = 5;  // Зеленый светодиод (двери открыты)
const int photoResistorPin = A0; // Аналоговый пин для фоторезистора

// Временной интервал, на который открываются двери (в миллисекундах)
const unsigned long doorOpenTime = 5000; // 5 секунд

// Переменные для управления состоянием
unsigned long doorOpenStartTime = 0;
bool doorIsOpen = false;

void setup() {
    // Инициализация пинов светодиодов как выходов
    pinMode(redLedPin, OUTPUT);
    pinMode(greenLedPin, OUTPUT);

    // По умолчанию двери закрыты (горит красный светодиод)
    digitalWrite(redLedPin, HIGH);
    digitalWrite(greenLedPin, LOW);

    // Инициализация последовательного порта для отладки
    Serial.begin(9600);
    Serial.println("Система автоматических дверей запущена");
    Serial.println("Двери закрыты");
}

void loop() {
    // Чтение значения с фоторезистора
    int sensorValue = analogRead(photoResistorPin);

    // Если значение превышает порог и двери еще не открыты
    if (sensorValue > 512 && !doorIsOpen) {
        openDoor();
    }

    // Если двери открыты и прошло достаточно времени
    if (doorIsOpen && (millis() - doorOpenStartTime >= doorOpenTime)) {
        // Проверяем значение фоторезистора снова
        sensorValue = analogRead(photoResistorPin);
    }
}
```

Рисунок 21 – Первая часть кода задачи №9


```

    if (sensorValue <= 512) {
        closeDoor();
    } else {
        // Обновляем время открытия, чтобы проверить снова через doorOpenTime
        doorOpenStartTime = millis();
        Serial.println("Двери остаются открытыми, человек все еще рядом");
    }
}

// Небольшая задержка для стабильности
delay(100);
}

// Функция для открытия дверей
void openDoor() {
    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);
    delay(5000);
    digitalWrite(greenLedPin, LOW);
    doorIsOpen = true;
    doorOpenStartTime = millis();
    Serial.println("Двери открываются!");
}

// Функция для закрытия дверей
void closeDoor() {
    digitalWrite(greenLedPin, LOW);
    digitalWrite(redLedPin, HIGH);
    delay(5000);
    digitalWrite(redLedPin, LOW);
    doorIsOpen = false;
    Serial.println("Двери закрываются!");
}

```

Рисунок 22 – Вторая часть кода задачи №9

Пример работы программы:

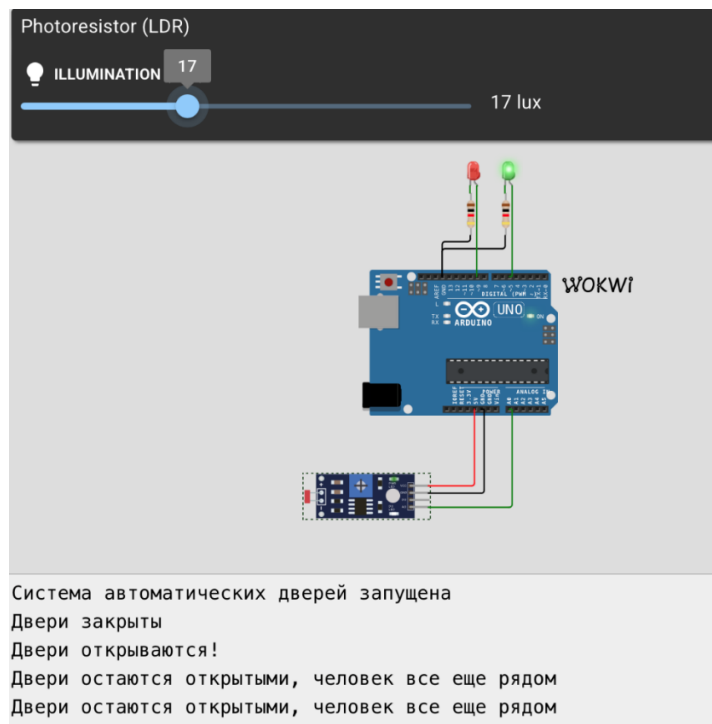


Рисунок 24 - Зеленый светодиод

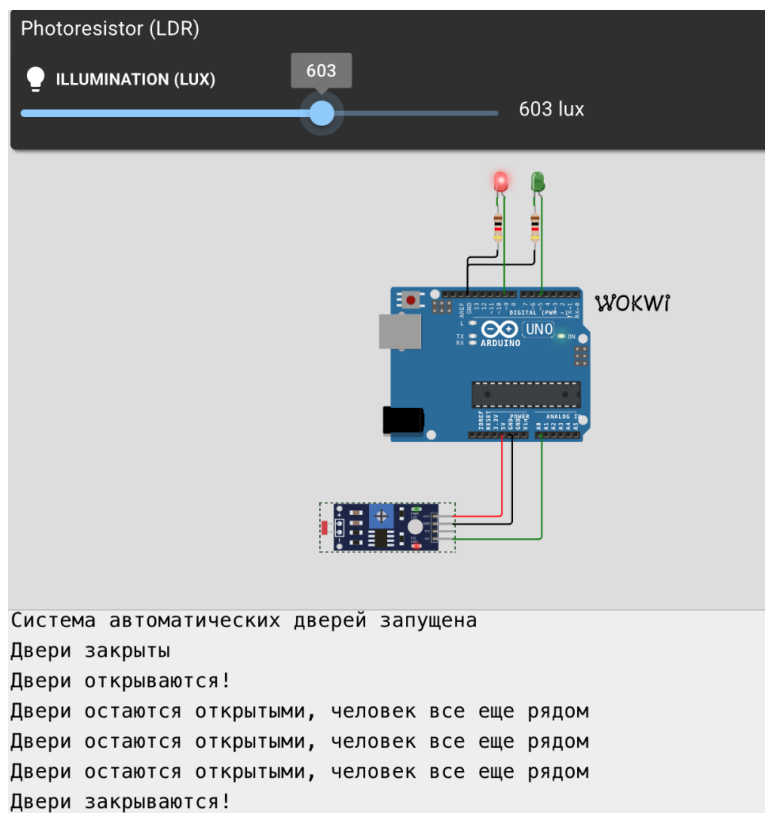


Рисунок 25 - Красный светодиод

Задание №10

Задачи:

- 1) Собрать схему подключения сервопривода
- 2) Написать программу для управления сервоприводом через последовательный порт

Логика работы программы:

запрашивается угол поворота сервопривода, если он отличен от того, на который повернут привод, то плавно повернуть до указанного. Программа работает в цикле, с возможностью постоянно изменять угол поворота.

Алгоритм программы:

- 1) *Инициализация:* программа подключает сервоприводом к пину 3, устанавливая его начальное положение на угол 0 градусов. Активируется коммуникация для приёма команд.
- 2) *Чтение угла:* в цикле программа постоянно проверяет наличие данных в порту. Если введено новое значение угла (от 0 до 180 градусов), и оно отличается от текущего угла сервопривода, привод плавно поворачивается на указанный угол.
- 3) *Плавный поворот:* Система вычисляет направление перемещения и последовательно изменяет угол с небольшим временным интервалом между шагами.
- 4) *Повторение:* Программа работает в непрерывном цикле, позволяя пользователю изменять угол поворота сервопривода в любой момент.

Код программы:

```
1  #include <Servo.h>
2
3  Servo myServo; // Создаем объект сервопривода
4
5  const int servoPin = 3; // Пин для подключения сервопривода
6  int currentAngle = 0;    // Текущий угол сервопривода (начальное положение)
7  int targetAngle = 0;    // Целевой угол сервопривода
8
9  void setup() {
10     myServo.attach(servoPin); // Подключаем сервопривод к указанному пину
11     myServo.write(currentAngle); // Устанавливаем начальное положение
12
13     Serial.begin(9600);        // Инициализация последовательного порта
14     Serial.println("Система управления сервоприводом готова");
15     Serial.println("Введите угол от 0 до 180 градусов:");
16 }
17
18 void loop() {
19     // Проверяем наличие данных в последовательном порту
20     if (Serial.available() > 0) {
21         String input = Serial.readStringUntil('\n'); // Читаем строку до символа новой строки
22         input.trim(); // Удаляем лишние пробелы
23
24         // Проверяем, является ли ввод числом
25         if (isValidNumber(input)) {
26             int newAngle = input.toInt(); // Преобразуем строку в число
27         }
```

Рисунок 25 – Первая часть кода задачи №10

```

28 // Проверяем, находится ли угол в допустимом диапазоне
29 if (newAngle >= 0 && newAngle <= 180) {
30     if (newAngle != currentAngle) {
31         targetAngle = newAngle;
32         Serial.print("Поворачиваю сервопривод на угол: ");
33         Serial.println(targetAngle);
34     } else {
35         Serial.println("Сервопривод уже в этом положении");
36     }
37 } else {
38     Serial.println("Ошибка: угол должен быть от 0 до 180 градусов");
39 }
40 } else {
41     Serial.println("Ошибка: введите числовое значение от 0 до 180");
42 }
43 }
44
45 // Плавное перемещение сервопривода к целевому углу
46 if (currentAngle != targetAngle) {
47     if (currentAngle < targetAngle) {
48         currentAngle++;
49     } else {
50         currentAngle--;
51     }

```

Рисунок 26 – Вторая часть кода задачи №10

```

44
45 // Плавное перемещение сервопривода к целевому углу
46 if (currentAngle != targetAngle) {
47     if (currentAngle < targetAngle) {
48         currentAngle++;
49     } else {
50         currentAngle--;
51     }
52     myServo.write(currentAngle);
53     delay(15); // Задержка для плавности движения
54 }
55 }
56
57 // Функция проверки, является ли строка числом
58 bool isValidNumber(String str) {
59     for (byte i = 0; i < str.length(); i++) {
60         if (!isDigit(str.charAt(i))) {
61             return false;
62         }
63     }
64     return str.length() > 0;
65 }

```

Рисунок 27 – Третья часть кода задачи №10

Пример работы программы:

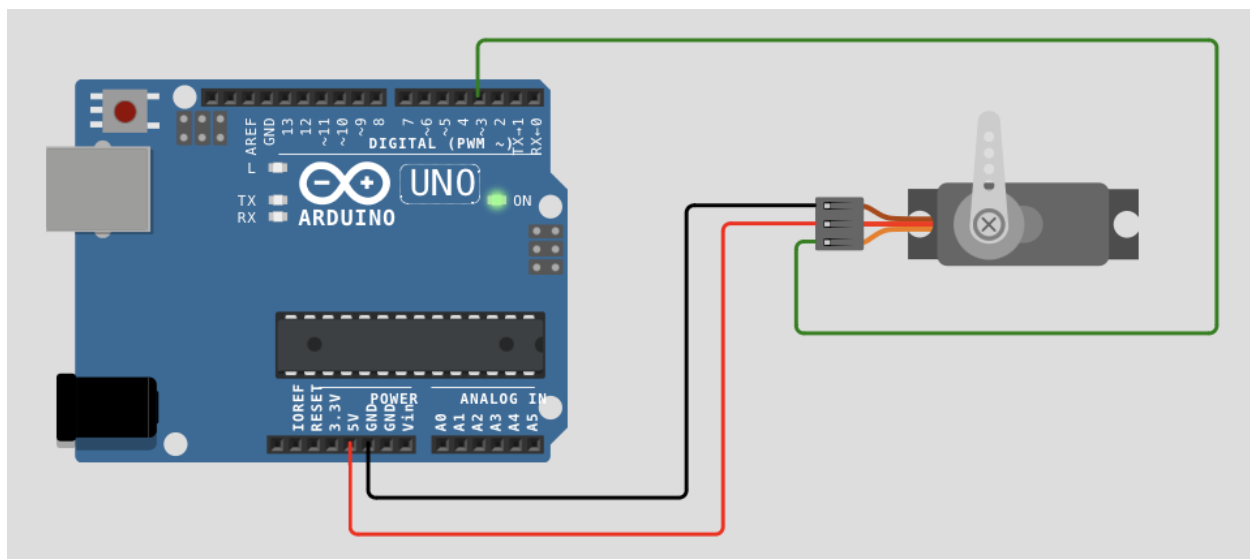


Рисунок 28 - Начальное положение на 0 градусов

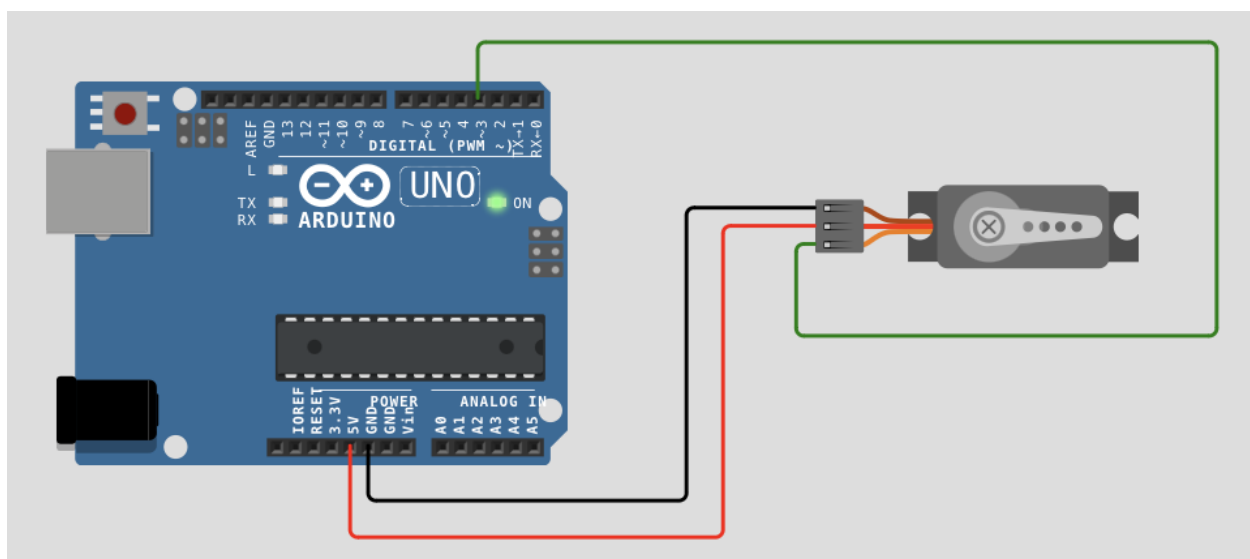


Рисунок 29 - Положение на 90 градусов

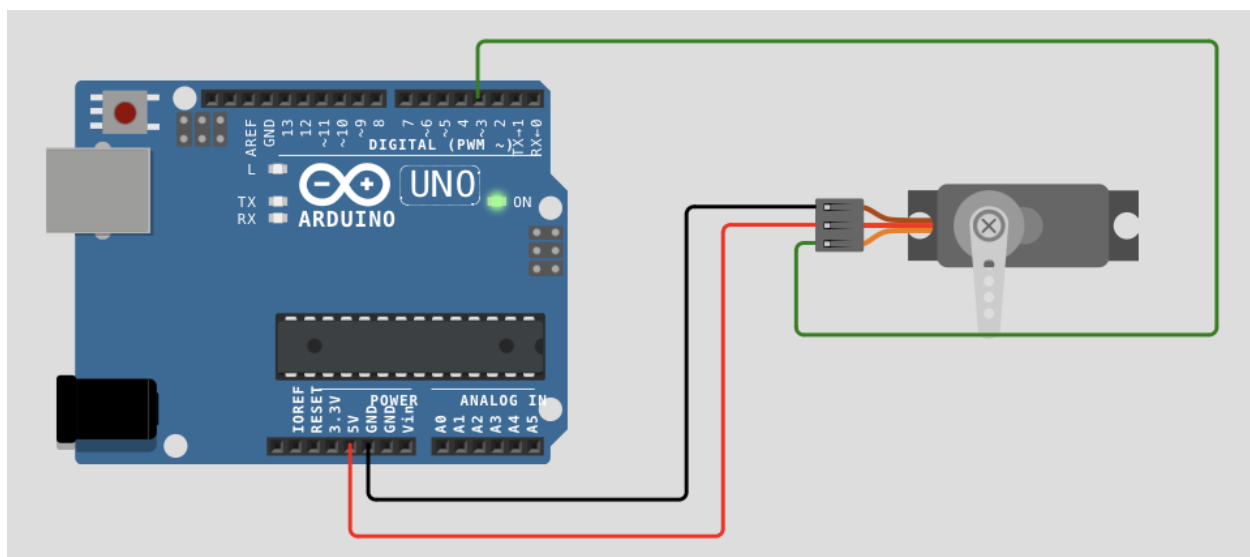


Рисунок 30 - Положение на 180 градусов

Задание №11

Задача:

Найдите все зеленые и желтые объекты на изображении. Найдите центры всех зеленых и желтых объектов. Выделите границы зеленых объектов синей рамкой. Выделите границы желтых объектов зеленой рамкой.

Алгоритм программы:

- 1) Загрузка изображения;
- 2) Применяем размытие Гаусса, чтобы убрать шум, и переводим изображение из BGR в HSV (чтобы было проще выделять цвета);
- 3) Задаем диапазон зеленого цвета в HSV, строим бинарную маску, в которой белые пиксели будут соответствовать зелёным областям;
- 4) Находим контуры на маске и иерархию вложенности (каждый контур соответствует зеленому объекту или его внутренним деталям);
- 5) Оставляем только внешние контуры (у которых нет родителя в иерархии);
- 6) Для каждого внешнего контура строим минимальную описывающую окружность, центр которой будет центром объекта;
- 7) На исходном изображении рисуем красную точку в центре каждого зеленого объекта;
- 8) Выводим результат.

Код программы:

```

1  import cv2
2  import numpy as np
3
4  # Загрузка изображения
5  image = cv2.imread("image.jpg")
6
7  if image is None:
8      print("Ошибка: изображение не загружено. Проверь путь к файлу.")
9      exit()
10
11  # Размытие
12  blurred = cv2.GaussianBlur(image, (11, 11), 0)
13
14  # Перевод в HSV
15  hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
16
17  # Диапазон зелёного цвета (при необходимости подстрой)
18  green_min = np.array([40, 70, 50], np.uint8)
19  green_max = np.array([80, 255, 255], np.uint8)
20
21  # Маска зелёных областей
22  green_mask = cv2.inRange(hsv, green_min, green_max)
23
24  # Контур и иерархия
25  green_contours, green_hierarchy = cv2.findContours(
26      green_mask.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE
27  )
28
29  # Обработка контуров
30  if green_hierarchy is not None:
31      for i, contour in enumerate(green_contours):
32          if green_hierarchy[0][i][3] == -1: # внешний контур
33              # Минимальная окружность
34              (x, y), radius = cv2.minEnclosingCircle(contour)
35              center = (int(x), int(y))
36
37              # Отметить центр красной точкой
38              cv2.circle(image, center, 3, (0, 0, 255), -1)
39
40  # Показ результата
41  cv2.imshow("Result", image)
42  cv2.waitKey(0)
43  cv2.destroyAllWindows()

```

Рисунок 31 – код задачи №11

Пример работы программы:

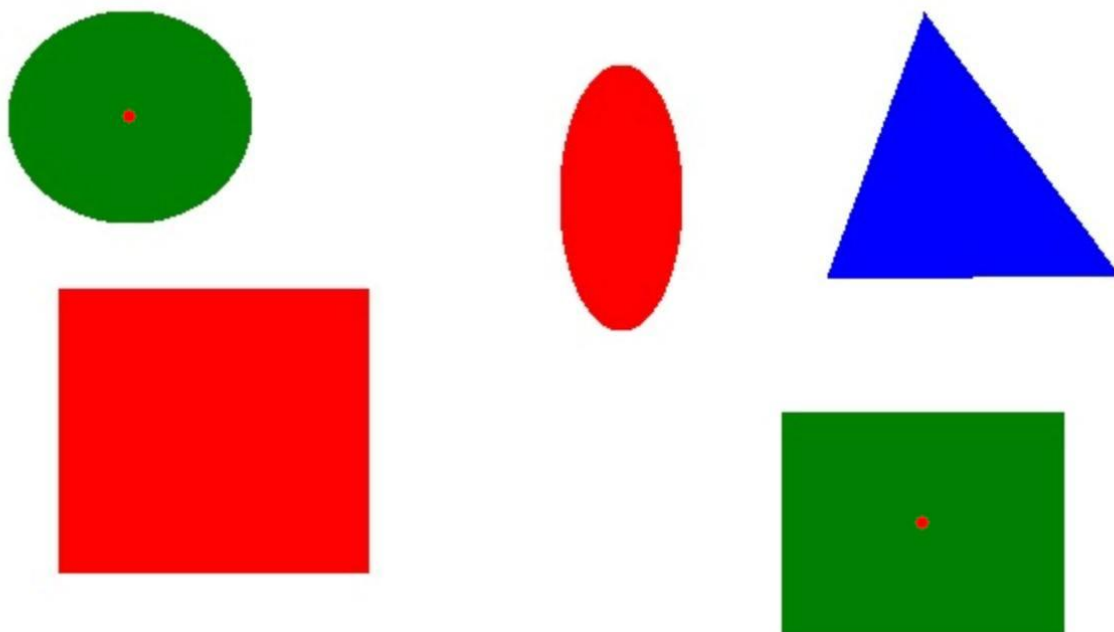


Рисунок 32 – Первый пример работы программы

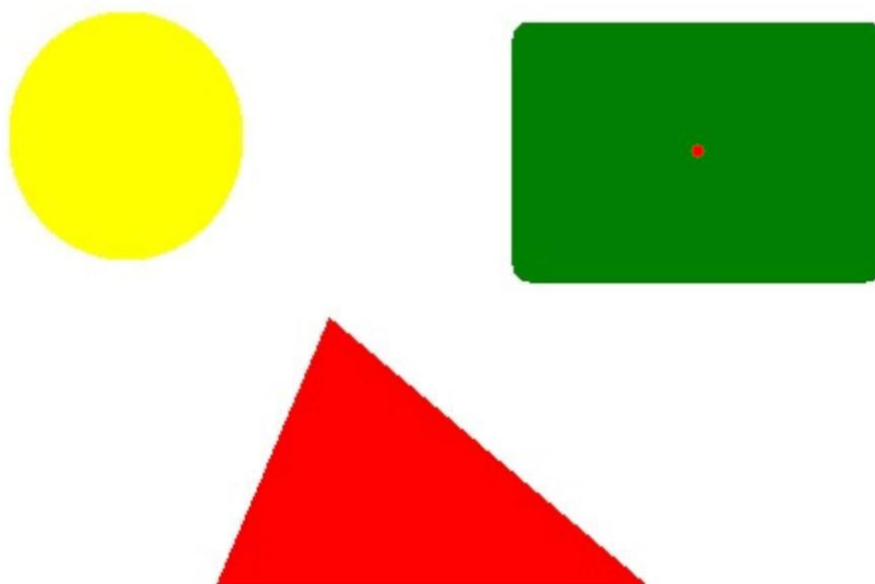


Рисунок 33 – Первый пример работы программы

Отзыв о посещении филиала АО «СО ЕЭС» Иркутское РДУ

В ходе учебной практики наша группа познакомилась с деятельностью Иркутского регионального диспетчерского управления (РДУ), являющегося филиалом АО «Системный оператор Единой энергетической системы». Данное предприятие осуществляет ключевые функции по поддержанию надежной работы энергосистемы Иркутской области. В сферу ответственности управления входит координация работы объектов генерации и электросетевого комплекса, а также оперативное управление режимами производства и передачи электроэнергии.

В процессе ознакомительной экскурсии было отмечено возрастающее влияние цифровизации на энергетическую отрасль. Предприятие располагает собственным подразделением информационных технологий, которое занимается разработкой и внедрением программных комплексов для автоматизации диспетчерской деятельности. К числу таких систем относится автоматизированный информационный комплекс (АИК), применяемый для выполнения расчетов и визуализации оперативной обстановки. Кроме того, внедряется перспективная имитационная модель энергосистемы («цифровой двойник»), предназначенная для моделирования поведения электростанции в различных сценариях.

Проведенное мероприятие позволило расширить представление о принципах диспетчерского управления в электроэнергетике, оценить роль IT-инфраструктуры в обеспечении его эффективности и осознать высокую степень ответственности, связанную с бесперебойным энергоснабжением Иркутской области.

Отзыв о посещении компании ISPsystem

В рамках учебной практики наша группа посетила компанию ISPsystem, специализирующуюся на разработке программного обеспечения для управления IT-инфраструктурой. Следует подчеркнуть, что деятельность организации сосредоточена исключительно на создании и совершенствовании собственных программных продуктов, без предоставления хостинговых услуг, заказной разработки или технической поддержки сторонних решений.

Сотрудники компании демонстрировали вовлеченность и заинтересованность в собственном деле и проекте. Произвели впечатление профессионалов, горящих своей сферой работы.

Были представлены ключевые направления, в которых компания заинтересована в привлечении специалистов: разработка программного обеспечения (включая Backend, Frontend, QA, проектный менеджмент, UX/UI-дизайн и аналитику), коммерческая деятельность (маркетинг и продажи), техническая поддержка, работа с документацией и локализацией, а также административно-хозяйственные функции. Таким образом, компания предлагает карьерные возможности для специалистов различных профилей.

Экскурсия оставила положительное впечатление о ISPsystem. Особого внимания заслуживают взаимоотношения внутри коллектива. Ощущаются дружелюбность и теплая атмосфера. Посещение ISPsystem продемонстрировало пример успешной IT-компании, где профессиональное развитие сотрудников гармонично сочетается с развитием продуктовой линейки.

Заключение

В ходе прохождения практики были успешно закреплены знания, полученные в рамках дисциплин «Информатика» и «Программирование». Особое внимание уделялось совершенствованию навыков программирования на языке C++, а также работе на языке Python и с аппаратной платформой Arduino и различными модулями, что помогло лучше понять принципы взаимодействия программного и аппаратного обеспечения и расширить свой спектр возможностей.

В ходе выполнения практического задания по машинному зрению были созданы программы на Python с использованием специализированных библиотек компьютерного зрения (OpenCV) и математических вычислений (NumPy). Это помогло освоить фундаментальные принципы обработки изображений и технологий распознавания цветных объектов.

В рамках практики мы также посетили две организации: **ISPsystem** и **филиал АО «СО ЕЭС» Иркутское РДУ**. Эти экскурсии позволили познакомиться с деятельностью современных IT- и энергетических компаний, их требованиями к сотрудникам и организации рабочего процесса.

Закрепление материала, освоение новых умений и возможность увидеть работу IT-специалистов изнутри помогли ближе познакомиться с будущей профессией и дали мотивацию стремиться к большему.

Список литературы

1. Wokwi Arduino Simulator: онлайн-симулятор Arduino-проектов [Электронный ресурс]. – URL: <https://wokwi.com/arduino>
2. AlexGyver: подробное руководство по работе со светодиодными лентами WS2812B [Электронный ресурс]. – URL: https://alexgyver.ru/ws2812_guide
3. AlexGyver: руководство по работе со светодиодными матрицами [Электронный ресурс]. – URL: https://alexgyver.ru/matrix_guide
4. AlexGyver: работа с аналоговыми входами Arduino [Электронный ресурс]. – URL: <https://alexgyver.ru/lessons/analog-pins>
5. AlexGyver: работа с последовательным портом Serial [Электронный ресурс]. – URL: <https://alexgyver.ru/lessons/serial>
6. AlexGyver: подключение и управление сервоприводами [Электронный ресурс]. – URL: <https://alexgyver.ru/lessons/servo>

ДНЕВНИК

прохождения практики

обучающегося Ефимова Полина Олеговна, ИСИ6-24-1
(фамилия, имя, отчество, группа)

курс 1

направление Информатика и вычислительная техника

профиль Интеллектуальные системы обработки
информации и управления

в ИРНИТУ
(наименование профильной организации)

Иркутск, 2025

Руководителем практики от структурного подразделения назначен:
Кононенко Роман Владимирович, доцент института ИТиАД
(ФИО, должность)

**Рабочий график (план) прохождения практической подготовки
(заполняется обучающимся)**

№ п/п	Период практики	Содержание выполненных работ	Подпись руководителя практики от структурного подразделения
1	16.06.2025	Решила задачу №1, Составила резюме на hh.ru и superjob.ru.	Tko
2	17.06.2025	Решила задачу №2, Решила задачу №3.	Tko
3	18.06.2025	Решила задачу №4, Решила задачу №5, Решила задачу №6.	Tko
4	19.06.2025	Решила задачу №7.	Tko
5	20.06.2025	Изучила теоретический материал для задачи №8, Изучила теоретический материал для задачи №9.	Tko
6	21.06.2025 – 23.06.2025	Решила задачу №8, Решила задачу №9. Изучила теоретический материал для задачи №10.	Tko
7	24.06.2025	Экскурсия в филиал АО Со Еэс Иркутское РДУ.	Tko
8	25.06.2025	Решила задачу №10, Изучила теоретический материал для задачи №11.	Tko
9	26.06.2025	Экскурсия в IT-компанию ISPsystem.	Tko
10	27.06.2025	Решила задачу №11.	Tko

Дата фактического прибытия
обучающегося в структурное подразделение 16.06.2025

Дата фактического убытия
обучающегося из структурного подразделения 28.06.2025

Руководитель образовательной программы Кононенко Р.В.
(ФИО, подпись)

Директор института Говорков А.С.
(ФИО, подпись)

Индивидуальное задание на прохождение
учебной практики: технологической (проектно-технологической)
практики

для Ефимовой Полины Олеговны
(ФИО обучающегося полностью)

обучающегося 1 курса группы ИСИБ-24-1
по направлению подготовки Информационные системы и технологии
профиль Интеллектуальные системы обработки информации и управления

Место прохождения практики: ИРНИТУ

Сроки прохождения практики с «16» июня 2025 г. по «29» июня 2025 г.

Цели и задачи прохождения практики:

Закрепление теоретических знаний, полученных в ходе изучения дисциплин «Информатика» и «Программирование», а также развитие практических навыков в области разработки программного обеспечения.

Содержание практики, вопросы, подлежащие изучению:

Совершенствование умений программирования на языке C++; работа с микроконтроллерной платформой Arduino и изучение принципов взаимодействия с различными модулями; освоение основ программирования на языке Python; знакомство с базовыми приёмами машинного зрения

Планируемые результаты практики:

Закрепление и углубление знаний по языку программирования C++; Приобретение навыков работы с Arduino; Формирование практических умений в области разработки программного обеспечения для управления устройствами; Получение первоначального опыта применения методов машинного зрения; Развитие навыков отладки, тестирования и документирования программного кода;

Руководитель практики от
института ИТиАД

Тео / Кононенко Р.В. /
(подпись)

Согласовано:

Руководитель ООП

Тео / Кононенко Р.В. /
(подпись)

« 16 » 06 2025 г.

С настоящим индивидуальным заданием и с программой практики
ознакомлен(а), задание принято к исполнению

Ефимов « » 2025 г.
(подпись)