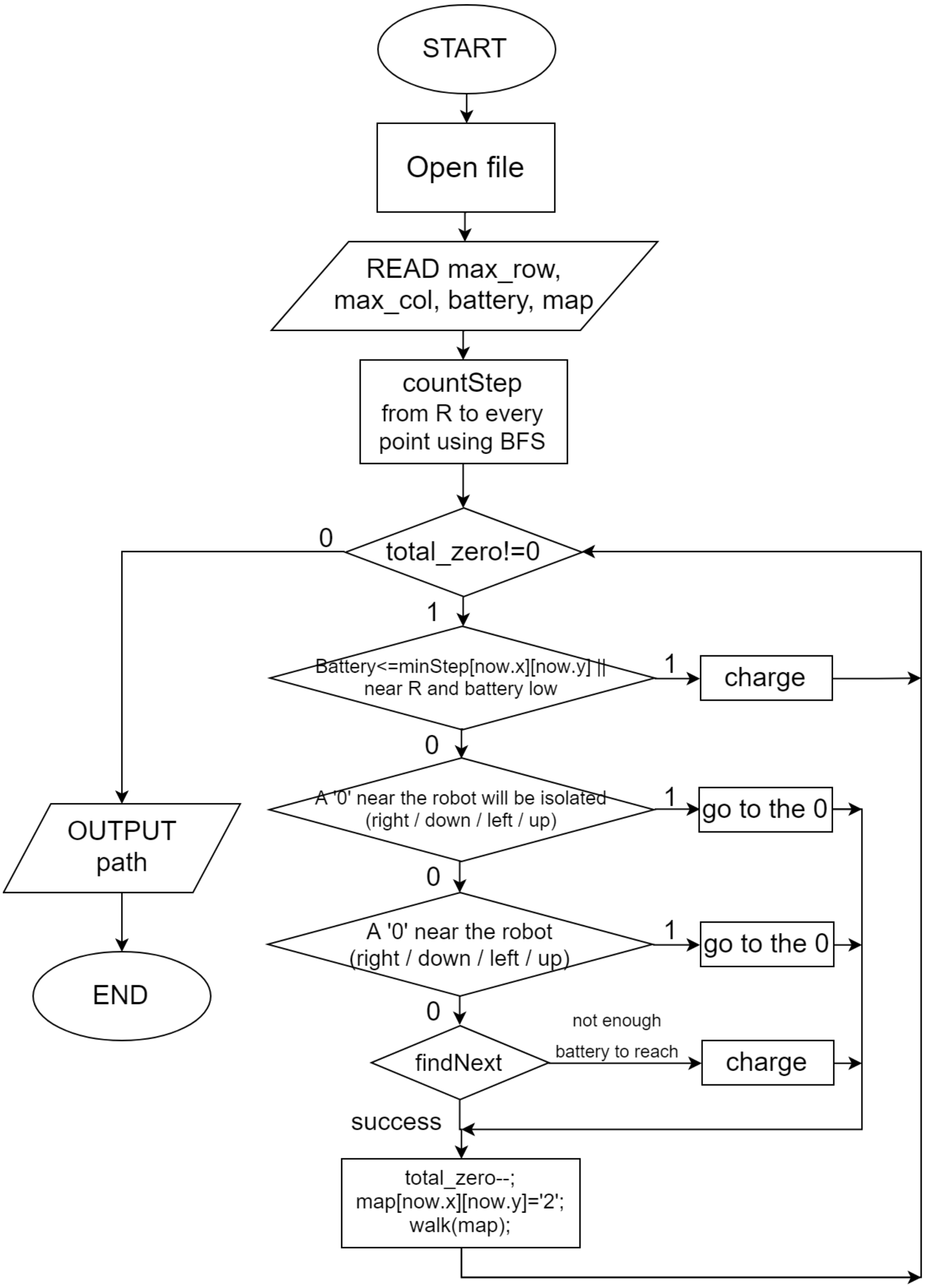
Project #2: Floor Cleaning Robot

1. Project Description

1.1 Program Flow Chart

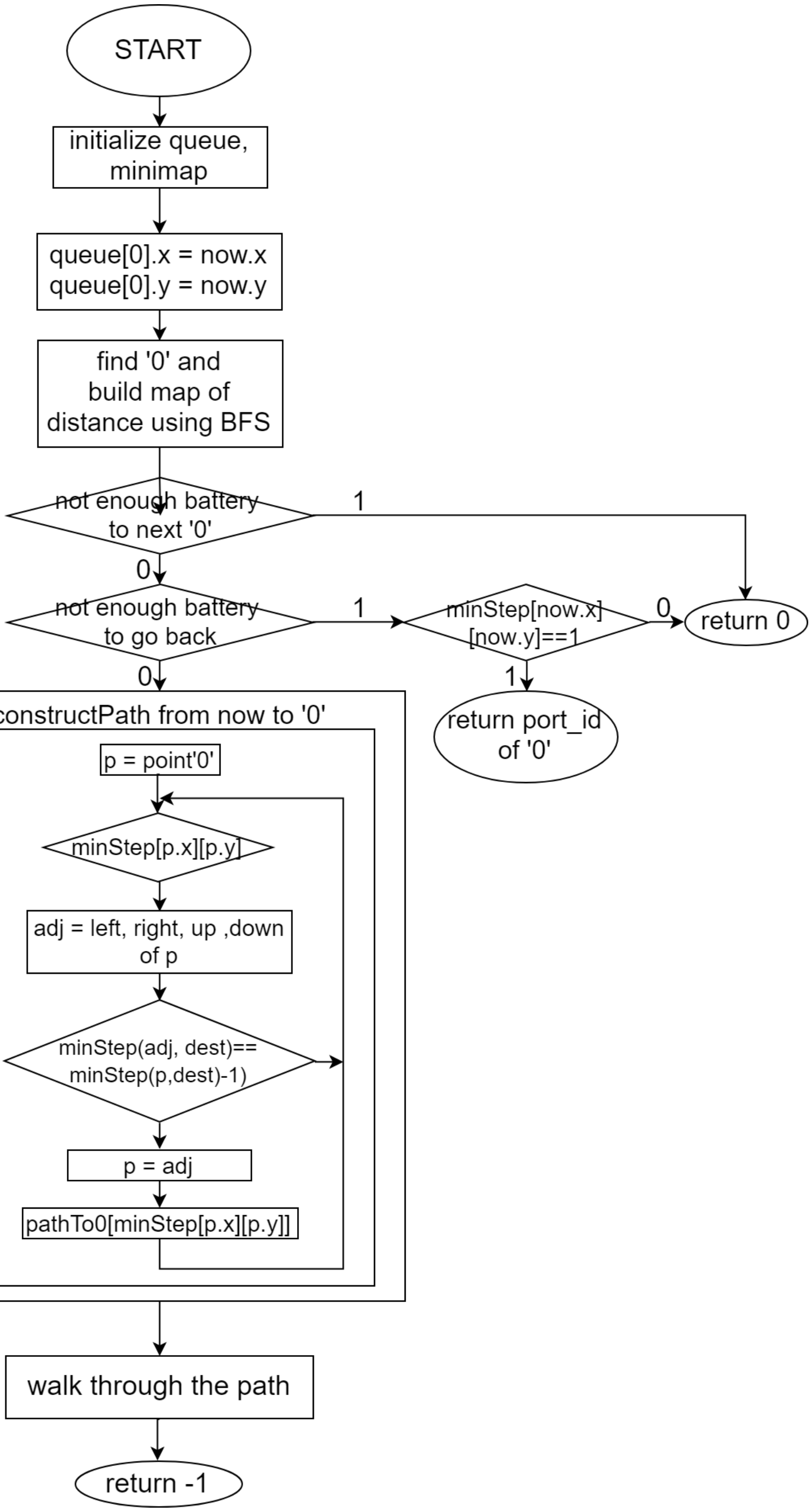
1. main()

First, build a matrix of distance from all point to ‘R’. Then, for each adjacent, point ‘up, down, left, right’, if it is ‘0’ then go to the adjacent point. Prioritize the point that will be isolated. If no adjacent point is ‘0’ then go to the nearest 0 by findNext, until total_zero become 0. If battery not enough then go to ‘R’ for charging using charge()


START
Open file
READ max_row, max_col, battery, map
countStep from R to every point using BFS
total_zero!=0
0
1
Battery<=minStep[now.x][now.y] || near R and battery low
1
charge
0
A '0' near the robot will be isolated (right / down / left / up)
1
go to the 0
0
A '0' near the robot (right / down / left / up)
1
go to the 0
0
findNext
not enough battery to reach
charge
success
total_zero--; map[now.x][now.y]='2'; walk(map);
OUTPUT path
END

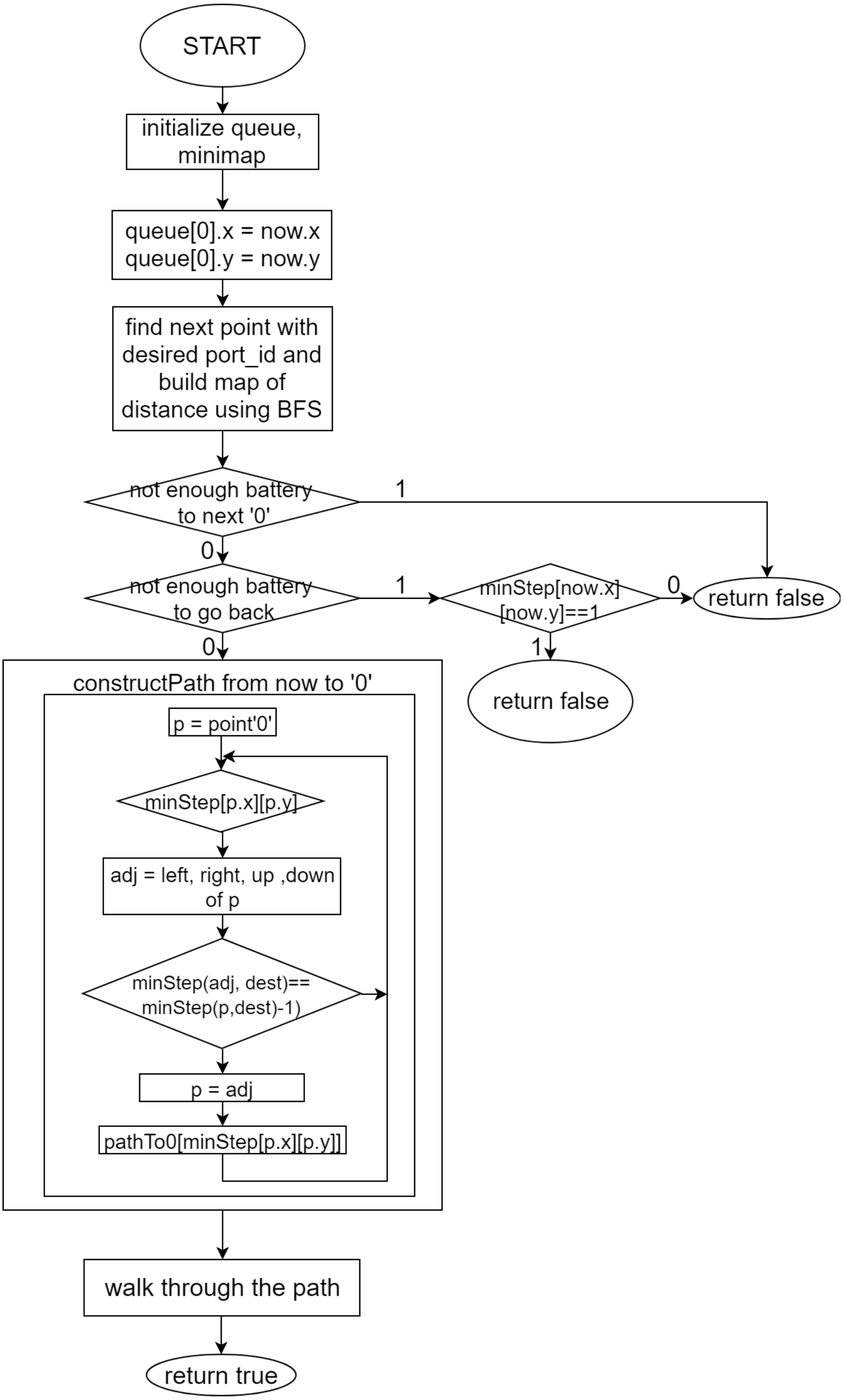
2. findNext() – use BFS

go to the next '0' element. If can't go to the element because of not enough battery, return 0. If the function is called by charge, return port_id of next point '0' found.

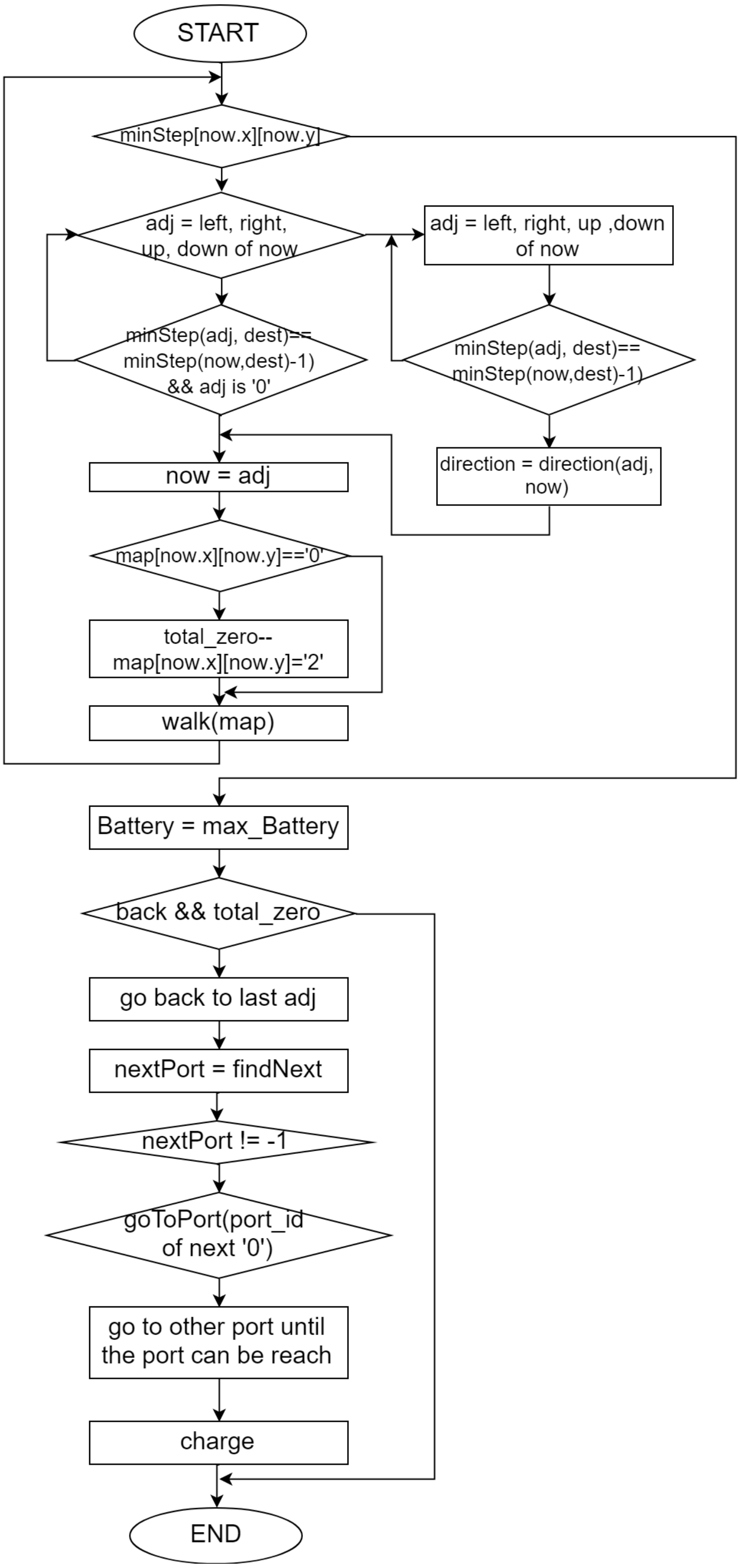

START
initialize queue, minimap
queue[0].x = now.x
queue[0].y = now.y
find '0' and build map of distance using BFS
not enough battery to next '0'
1
0
not enough battery to go back
1
minStep[now.x][now.y]==1
0
return 0
1
return port_id of '0'
0
constructPath from now to '0'
p = point'0'
minStep[p.x][p.y]
adj = left, right, up ,down of p
minStep(adj, dest)== minStep(p,dest)-1)
p = adj
pathTo0[minStep[p.x][p.y]]
walk through the path
return -1

3. goToPort() – use BFS

goToPort() is similar to findNext(), but its function is to go to a point with port_id of desired point instead.


START
initialize queue, minimap
queue[0].x = now.x
queue[0].y = now.y
find next point with desired port_id and build map of distance using BFS
not enough battery to next '0'
1
0
not enough battery to go back
1
minStep[now.x][now.y]==1
0
return false
1
return false
0
constructPath from now to '0'
p = point'0'
minStep[p.x][p.y]
adj = left, right, up ,down of p
minStep(adj, dest)== minStep(p,dest)-1
p = adj
pathTo0[minStep[p.x][p.y]]
walk through the path
return true

4. charge()
   Go back to ‘R’ and then go to next 0 element by findNext(). If can’t reach point ‘0’ from the incoming port, a port_id of nearest point ‘0’ will be return. Then switch port until the point can be reached.


START
minStep[now.x][now.y]
adj = left, right, up, down of now
adj = left, right, up ,down of now
minStep(adj, dest)== minStep(now,dest)-1) && adj is '0'
minStep(adj, dest)== minStep(now,dest)-1)
direction = direction(adj, now)
now = adj
map[now.x][now.y]=='0'
total_zero-- map[now.x][now.y]='2'
walk(map)
Battery = max_Battery
back && total_zero
go back to last adj
nextPort = findNext
nextPort != -1
goToPort(port_id of next '0')
go to other port until the port can be reach
charge
END

1.2 Handling testcases

1. Use `argv`, `argc` to get the `input_student_id`
   Reference: https://stackoverflow.com/questions/3024197/what-does-int-argc-char-argv-mean

```
int main (int argc, char* argv[]) {
```

2. Expand the `argv` to get the directory of testcase
3. Open files as `ifstream` and `ofstream`
   Reference: http://www.cplusplus.com/doc/tutorial/files/
   http://www.cplusplus.com/reference/fstream/ifstream/
   http://www.cplusplus.com/reference/fstream/ofstream/
   **Error message is designed to be shown when unable to open testcase

```
    ifstream input(dir_in);
    ofstream output(dir_out);

    if(input.is_open() && output.is_open()) {
        /* Code */
        input.close();
        output.close();
    }

    else if(!input.is_open()) cout << "Unable to open testcase" << endl;
    else cout << "Unable to open output" << endl;
```

2. Testcase Design

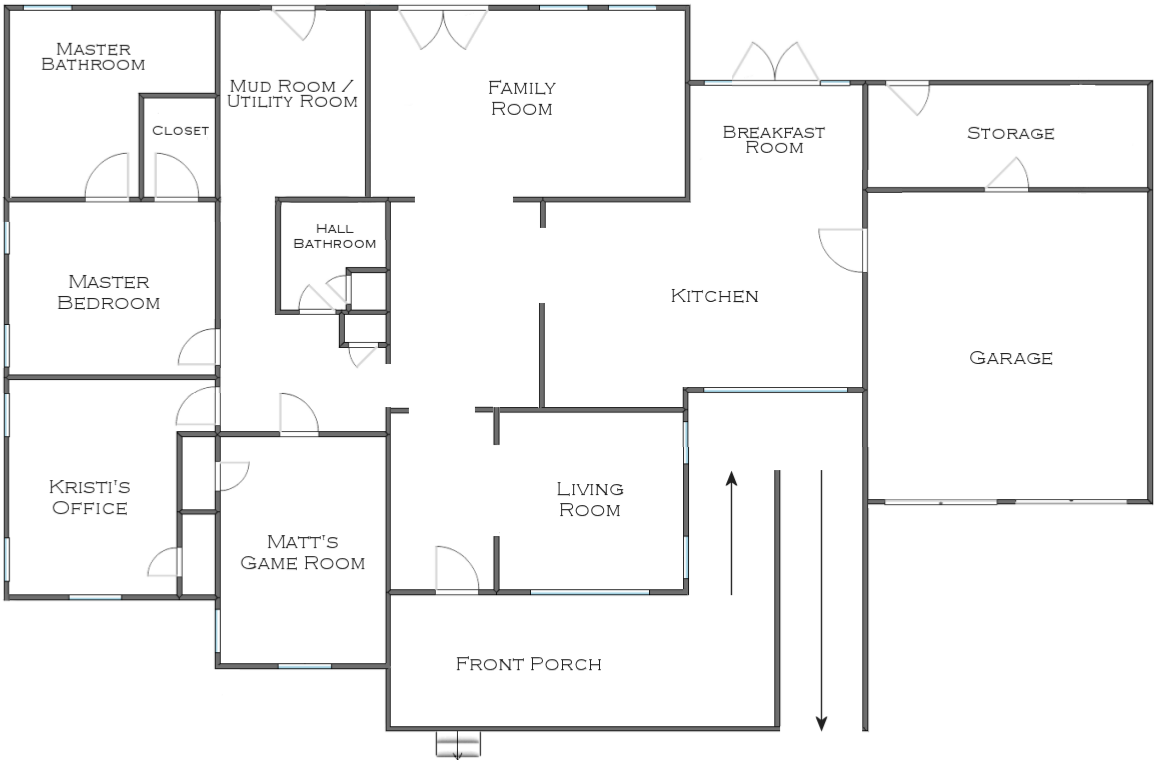
2.1 Design Concept

I would like to design a more real house-like testcase. So I find a house floor plan and convert it into a matrix.

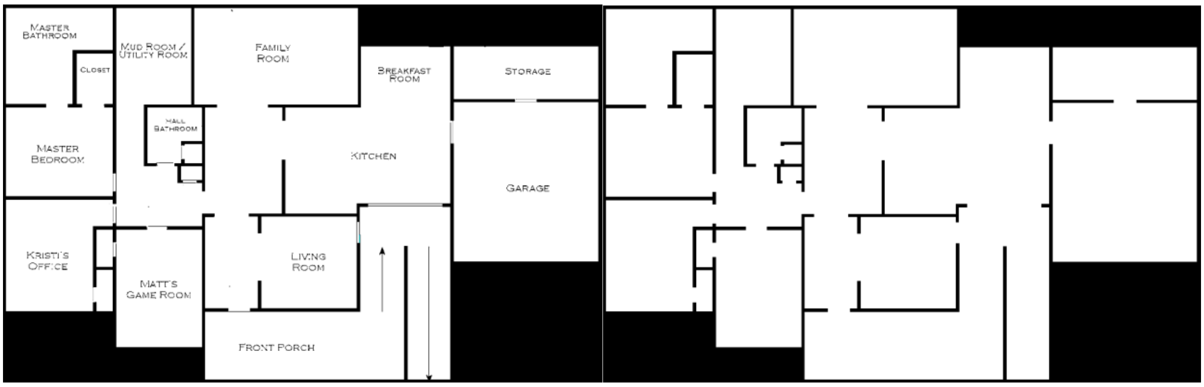
The size of matrix is small, which is 189*300.

2.2 Procedure

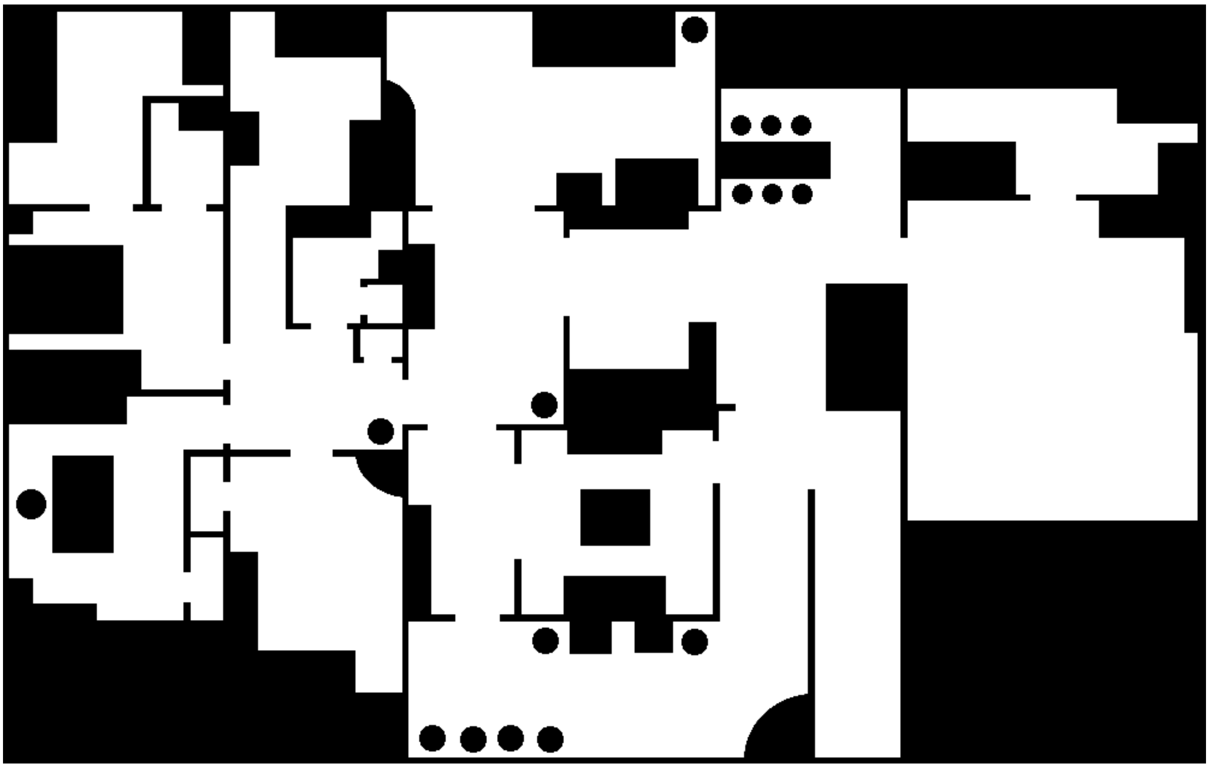
1. Google a house floor plan



2. Use Photoshop to redraw the walls



3. Add furniture



4. Use the website below to turn image into c-array according to its RGB value of each pixel: https://littlevgl.com/image-to-c-array

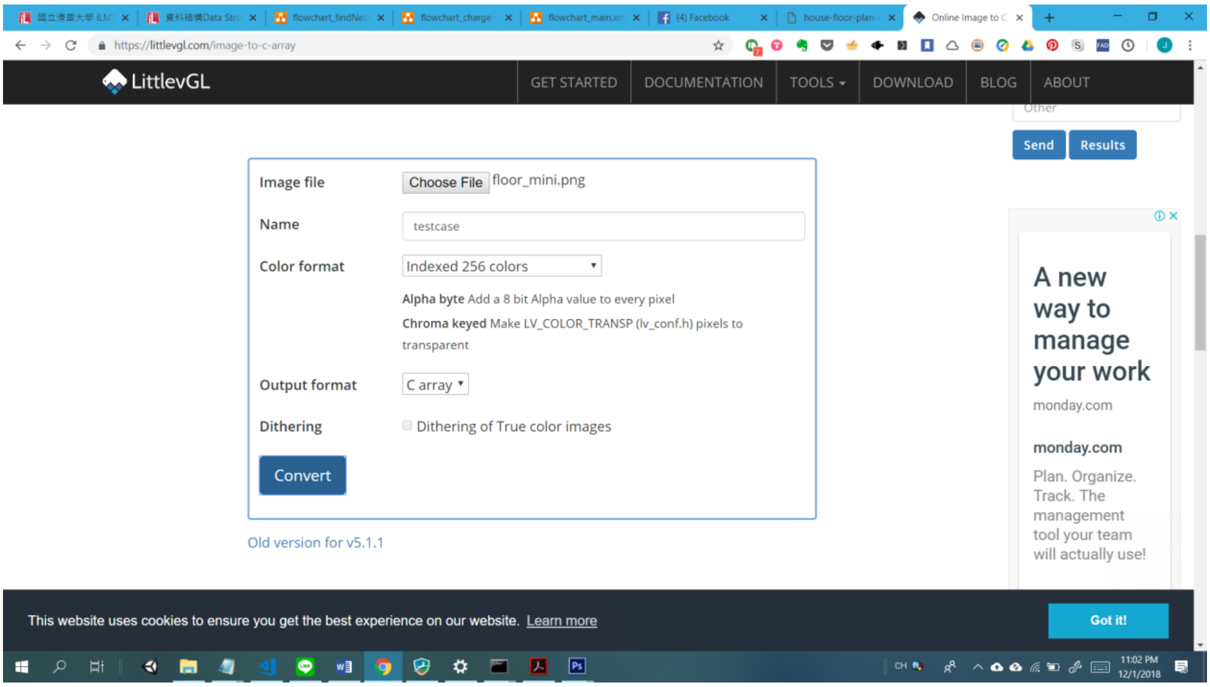


Image file: Choose File floor_mini.png

Name: testcase

Color format: Indexed 256 colors

Alpha byte Add a 8 bit Alpha value to every pixel

Chroma keyed Make LV_COLOR_TRANSP (lv_conf.h) pixels to transparent

Output format: C array

Dithering: □ Dithering of True color images

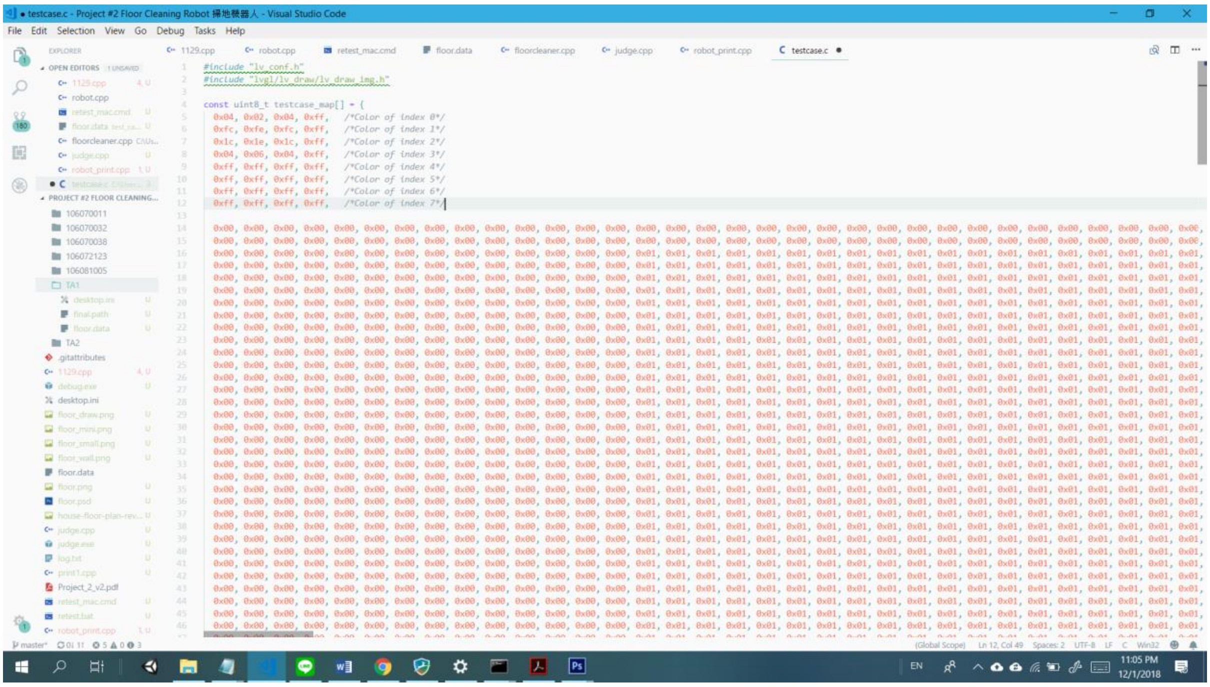
Convert

Old version for v5.1.1

This website uses cookies to ensure you get the best experience on our website. Learn more

Got it!

5. Output:



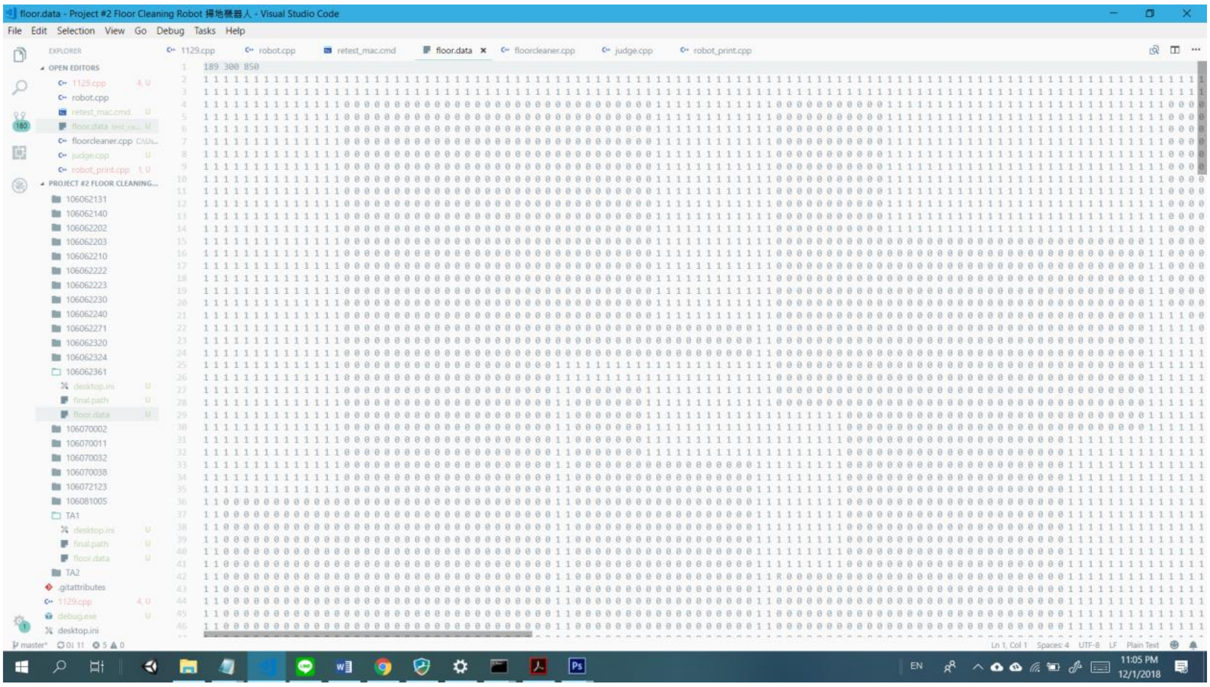
```
#include "lv_conf.h"
#include "lvgl/lv_draw/lv_draw_img.h"

const uint8_t testcase_map[] = {
  0x04, 0x02, 0x04, 0xff,   /*Color of index 0*/
  0xfc, 0xfe, 0xfc, 0xff,   /*Color of index 1*/
  0x1c, 0x1e, 0x1c, 0xff,   /*Color of index 2*/
  0x04, 0x06, 0x04, 0xff,   /*Color of index 3*/
  0xff, 0xff, 0xff, 0xff,   /*Color of index 4*/
  0xff, 0xff, 0xff, 0xff,   /*Color of index 5*/
  0xff, 0xff, 0xff, 0xff,   /*Color of index 6*/
  0xff, 0xff, 0xff, 0xff,   /*Color of index 7*/

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

6. Convert in to ‘0’ or ‘1’ using replace tool

```
“0xff, ” -> “0 ”
Other -> “1 ”
```



```
189 300 850
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
```

7. Correct the format and save as floor.data