AI Project 9 – Comp 415



AI Documentation

Date: December 2019

Table of Contents

[Introduction 3](#_Toc28106924)

[Team Members Delegations 3](#_Toc28106925)

[Tasks and contributions 3](#_Toc28106926)

[Table Key 3](#_Toc28106927)

[Project Resources and Components 3](#_Toc28106928)

[Tools used. 3](#_Toc28106929)

[Libraries used 4](#_Toc28106930)

[Implementation of code & Design 4](#_Toc28106931)

[Mile Stones and Setbacks 9](#_Toc28106932)

[Mile Stones 9](#_Toc28106933)

[Setbacks 9](#_Toc28106934)

[Hardware Implementation 9](#_Toc28106935)

[Team members Acceptance 13](#_Toc28106936)

# Introduction

In this project we were given a text file called problem.txt containing available points on a 12x 12 grid. The text file also contained the number of start and end points that for the paths that we need to calculate using the Uniform Cost Algorithm.

# Team Members Delegations

Here we find the break-down of the project into its main parts as well as the team members who were involved on those parts. We have also shown the interest and commitment to each part of this project by the team members in the table.

Tasks and contributions

The below table is a distribution of tasks through the group

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Luca | Yakup | Dexter | Ümmet | Hassnain |
| Text File Parsing | LI | MC | LI | LI | HI&I |
| AI Implementation | MC | MC | LI | LI | LI |
| AI Coding | MC | MC | LI | LI | LI |
| Hardware Constructuion | MC | LI | HI&I | SI | LI |
| Hardware Coding | MC | SI | HI&I | HI&I | LI |
| Documentation | MC | SI | SI | SI | SI |
| Signature |  |  |  |  |  |

## Table Key

Below is the key for the effort and time put in by team members.

|  |  |
| --- | --- |
| Main Contributors | MC |
| Highly Involved and Interested | HI&I |
| Some Involvement | SI |
| Little to no involvement | LI |

# Project Resources and Components

## Tools used.

This main part of this project was implemented using C++. It was used to read in the problem text file and parse the information from the file into memory easy due to the high competency and experience of our team members using it. C++ was also used to implement the main Uniform Cost Algorithm as it makes the project run faster and more is very efficient in memory management.

For the hardware part of the project we used an Arduino Uno with a 64x128 LCD display. The software for the hardware was coded using the C programming language and the desired library for the LCD display (u8glib.h).

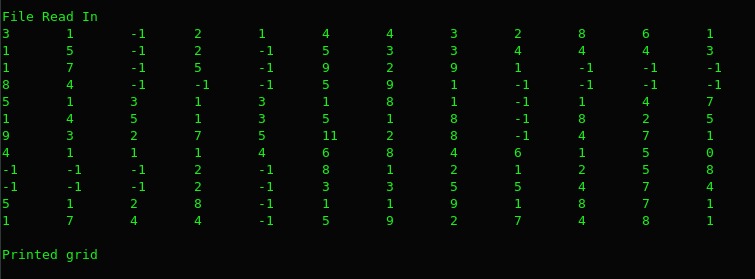
## Libraries used

In C++ we used the STL library (Which is a standard C++ library) as it has many useful functions such as sorting, lambda functions and vector objects that make implementing the uniform cost search easier and helps us spend less time debugging and testing as they are well defined and very efficient.

For the Hardware we had to use the desired library for the LCD display (U8glib.h) which allowed us to integrate our C code with the LCD display.

# Implementation of code & Design

In order to calculate the shortest path, we had to first read in the txt file and store it in memory.

Below is the output of our read in table in memory.

The code below is the function we used to read in our text file into memory.

void readFormattedFile() {

initialize();

std::ifstream problem; problem.open("Resources/problemformatted.txt");//Directory WILL CHANGE

if (problem.is\_open()) {

std::string path, readed;

std::cout << "File Open" << std::endl;

Cell back;

int count = 0;

while (problem >> readed) {

if (count == 0)

path = readed;

else if (count == 1)

workingtime = std::stoi(readed);

else {

std::string v, line;

std::getline(problem, line);

std::stringstream check(line);

int elimination = 0;

Cell starts;

Cell ends;

while (std::getline(check, v, ' ')) {

if (elimination == 1)

starts.x = std::stoi(v);

else if (elimination == 2)

starts.y = std::stoi(v);

else if (elimination == 3)

ends.x = std::stoi(v);

else if (elimination == 4)

ends.y = std::stoi(v);

elimination++;

}

Start.push\_back(starts);

End.push\_back(ends);

}

count++;

}

// parcing

int check = 0;//check=0 open,1close,2comma,3 ;

std::string num;

for (int i = 0; i < path.size();) {

if (path[i] == '(') {

check = 0;

i++;

}

else if (path[i] == ')') {

check = 1;

int index = find(back);

if (index != -1)

gridPoints[index] = back;

num.clear();

check = 0;

i++;

}

else if (path[i] == ',') {

check = 2;

i++;

}

else if (path[i] == ';') {

check = 3;

i++;

}

else {

int oldcheck = check;

check = 4;

num.push\_back(path[i]);

int j = i + 1;

while (path[j] != ')'&&path[j] != '('&&path[j] != ','&&path[j] != ';') {

num.push\_back(path[j]);

j++;

}

if (oldcheck == 0) {

back.x = std::stoi(num);

num.clear();

}

else if (oldcheck == 2) {

back.y = std::stoi(num);

num.clear();

}

else if (oldcheck == 3) {

back.cost = std::stoi(num);

num.clear();

}

else

num.clear();

i++;

}

}

std::cout << "Grid Created!" << std::endl;

for (auto &x : gridPoints) {

std::cout << "[" << x.x << "] [" << x.y << "] cost = " << x.cost << std::endl;

}

setPath();

std::cout << "Path Created!" << std::endl;

}

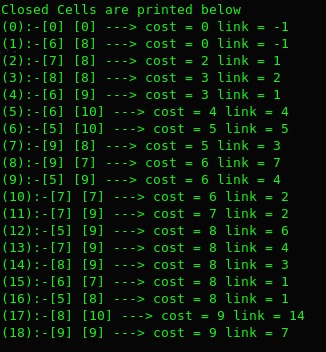
else

std::cout << "cannot open";

}

Our next task was to calculate the path from the start to end point using the Uniform Cost Algorithm to display everything to the screen.

The picture below shows our closed vector displaying the current paths and where which points they are connected to.



The code below is the function we used to calculate the shortest part. It works by checking the left, right, top and bottom cells of the current cell and adds them to the open vector which is sorted based on the current cost. The current element is then removed from the open vector and moved into the closed vector with a link to which element it came from, -1 being the start node, The element at the back of the open vector (the least cost element of the path) is then read in and checked in all directions. The process is repeated till the last element in the closed vector has the same value as the end point.

void path(Cell start, Cell end){

int index;

Cell temp(start.x,start.y,0,-1), addTemp(0,0,0,-1);

addOpen(start);

addClose(addTemp);

while(!((close.back().x == end.x) && (close.back().y == end.y))){

temp = pop();

addClose(temp);

index = findInClose(temp);

if(findInClose(move(temp,Right)) == -1){

addTemp = move(temp,Right);

addTemp.cost += temp.cost;

addTemp.link = index;

addOpen(addTemp);

}

if(findInClose(move(temp,Left)) == -1){

addTemp = move(temp,Left);

addTemp.cost += temp.cost;

addTemp.link = index;

addOpen(addTemp);

}

if(findInClose(move(temp,Up)) == -1){

addTemp = move(temp,Up);

addTemp.cost += temp.cost;

addTemp.link = index;

addOpen(addTemp);

}

if(findInClose(move(temp,Down)) == -1){

addTemp = move(temp,Down);

addTemp.cost += temp.cost;

addTemp.link = index;

addOpen(addTemp);

}

}

}

Once the closed vector is created, we backtrack from the last element in the vector to the beginning element containing the link address of -1. The code for backtracking is below.

void backtracking() {

auto open = grid.GetClose();

int i = open.size() - 1;

while (open[i].link != -1) {

shortestPath.push\_back(open[i]);

i = open[i].link;

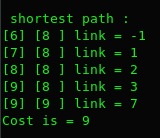
}

shortestPath.push\_back(open[i]);//for start from the starting point

std::reverse(shortestPath.begin(), shortestPath.end());//because our vectoro is working like stack our starting point end of the vector.

}

Our path is then output to a file containing the path and total cost in an array format. The image below shows the final path stored in memory before being sent to the output file.



# Mile Stones and Setbacks

## Mile Stones

Our Milestones for this project were small but were significant in allowing us to fully grasp the concept of the project.

Initially we had difficulty trying to understand how we would implement the Uniform Cost Algorithm. Our problem was we didn’t know how to store our points into memory and back track the path. We eventually solved this problem by using a stack to store all our points in memory and have an additional variable that will store the index from where the element came from called link.

Our next big Milestone was when we realised that we could display the grid on our 128x64 LCD screen using an enum class in C instead of a bitmap which was very difficult and complex to calculate.

## Setbacks

The setbacks from the project mainly came from the project question itself which read in the points and their weight in a very unfriendly way making it very difficult to parse the info from the text file into memory.

Another set back we had was one of our team members was unable to help on their respective tasks in the project due to health Issues. This meant other team members had to pick up the slack for the uncompleted tasks, putting more pressure on the 2 Team members who already did a lot of the work.

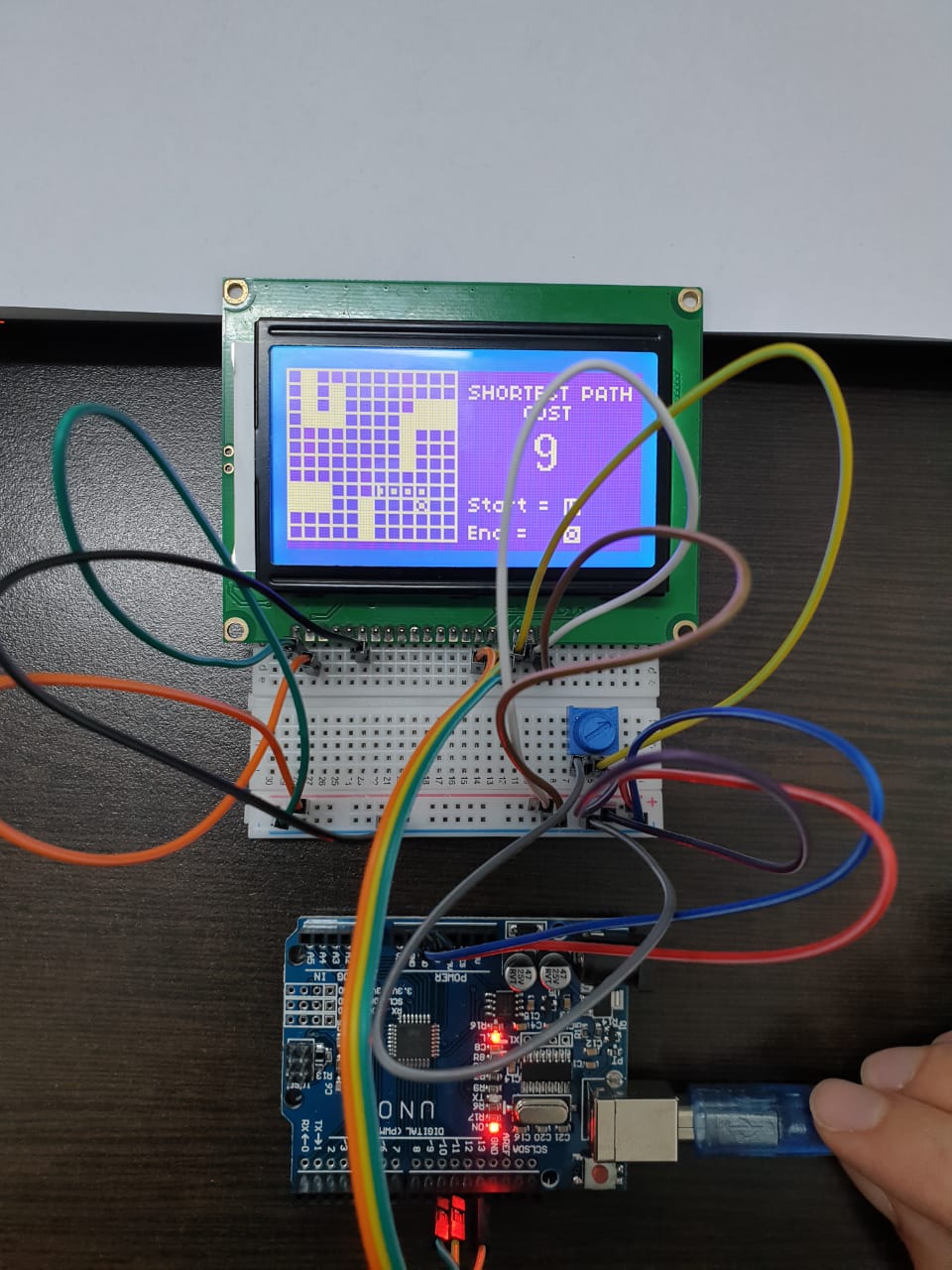
# Hardware Implementation

To implement the hardware of this project we needed the following components.

* Arduino Uno
* 64x128 LCD display board
* Wires to connect pins between the Arduino and the display board
* Electric bread board

With these components we were able to make a simple device for displaying the shortest path to the user through the display.

The picture below shows which pins are connected between the LCD and the Arduino using the bread board.



This hardware is run by the C code below, which creates a map that is created by our C program that we copy into the source code. We also have a points Array which holds the X and Y axis of the points to the shortest path. Our code then updates the Map from the Points Array to show the path on the map before displaying it to the screen.

The picture below is what the final Output screen will look like.



This is the code used to create the Point Array and Map that we got from the output of our C++ Program.

struct Point{

int x;

int y;

};

Point points[] = {

{6,8},{7,8},{8,8},{9,8},{9,9}

};

enum CellType{

W, //Wall

B, //Block

P, //Path

SR, //Start RIGHT Facing

SL, //Start LEFT Facing

SU, //Start UP Facing

SD, //Start Down Facing

E //End

};

CellType Cells[12][12]= {

{B,B,B,B,B,B,B,B,W,W,B,B},

{W,W,W,W,B,B,B,B,W,W,B,B},

{B,B,B,W,B,B,B,B,W,W,B,B},

{B,W,W,W,B,B,B,B,B,B,B,B},

{B,B,B,B,B,B,B,B,B,B,B,B},

{B,B,B,B,B,B,B,B,B,W,W,W},

{B,B,B,B,B,B,B,B,B,B,B,B},

{B,B,B,B,B,B,B,B,B,B,B,B},

{B,B,B,W,W,W,W,B,B,B,B,B},

{B,B,W,W,B,B,B,B,B,B,B,B},

{B,B,W,W,B,B,B,B,B,B,B,B},

{B,B,W,W,B,B,B,B,B,B,B,B},

};

This part of the code is what we used to change the values on the current map into the map with the path that we specified.

for(int i = 0; i < sizeof(points)/sizeof(Point); i++){

Point p = points[i];

if(i == 0){

Cells[p.x][p.y] = startDir(p,points[i+1]);

}

else if(i == sizeof(points)/sizeof(Point)-1){

Cells[p.x][p.y] = E;

}

else

Cells[p.x][p.y] = P;

}

After the new map has been created we then print it to the screen in the loop function as shown below.

void loop(void) {

u8g.firstPage();

do{

for(int y = 0; y < 12; y++){

for(int x = 0; x < 12; x++){

if(Cells[x][y] == B)

Block(x,y);

else if(Cells[x][y] == W)

Wall(x,y);

else if(Cells[x][y] == P)

Path(x,y);

else if(Cells[x][y] == SR)

Start\_RIGHT(x,y);

else if(Cells[x][y] == SD)

Start\_DOWN(x,y);

else if(Cells[x][y] == SL)

Start\_LEFT(x,y);

else if(Cells[x][y] == SU)

Start\_UP(x,y);

else if(Cells[x][y] == E)

End(x,y);

}

}

drawHeading();

Cost();

drawKey();

}while(u8g.nextPage());

delay(2000);

clearLCD();

}

We also print a key to the screen with the Total cost of the path as well as the start and end point tokens so that the user won’t get confused on what the start and end point is.

# Team members Acceptance

Approved by the Team Members:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Luca Tonini 164452

Team Manager

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yakup Beyoglu 163049

Team Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dexter Chipango 154304

Team Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ümmet Burak 150953

Team Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hassnain Nisar 164393

Team Member