	Benha University	**Grade:** Third year	Program Logo
	Benha Faculty of Engineering	**Project:** OUZO Game	
	Communication and Computer Engineering **Program**	**Date:** 18/5/2023	
	Second Semester	**Due to Date:** 18/5/2023	
	Course Title: Data structure and algorithms with C++	**Total Mark:** **marks**	
	Course Code: E1234	**Measured CLO:** CLO1,CLO2,CLO3	
	Instructor: D./ Eman Salem		

Project:

OUZO Game

Prepared by:

Moustafa Prince ELHag

Noran Nabil Ahmed

Nagham Nasser Kamal

Supervisor:

D./Eman Salem

Benha Faculty Of Engineering

Communication and Computer Department

May 2023

Table of Contents

Table of Figures

Abstract

The data structure is a tool to organize and manipulate the data. Data Structures enable the programmers to handle the data in an efficient way to save time and storage resources. For these resources, we use a kind of data sorting algorithm such a stack and linked list to implement our project. The stack is implemented based on the linked list to save the maximum storage as much as we can. As, the linked list depends on the dynamic memory allocation. Ouzo game uses the stack to contain two stages. Each stage has three levels. When the user win one of each stage, he will win the whole game.

Introduction

Ouzo prints a pattern of open brackets and symbols and the user must close these brackets and symbols in correct way. The game has 3 levels. In level 1, we print on lcd 4 symbols only and the user must close them. In level2, 6 symbols. In level3, 8 symbols. OUZO has easy and hard levels. In easy level, the cursor is moving up and down only and the user must enter correct closing symbols in the correct line. In hard level, the cursor moves up and down and to the next cell making the screen to move and the pattern will disappear if the user doesn't enter the symbols quickly and he must remember the symbols to be able to close it correctly. In this game, User has 3 trials only and after them, he will get game over. If the user can be lucky for 3 levels and win in all of them, he will win the game and there will be winner word on the lcd.

Component

Ouzo game has been implemented by some tools:

1- Arduino UNO as shown in figure 1.
2- Keypad as shown in figure 2.
3- LCD as shown in figure 3.

Figure 1 shows arduino UNO

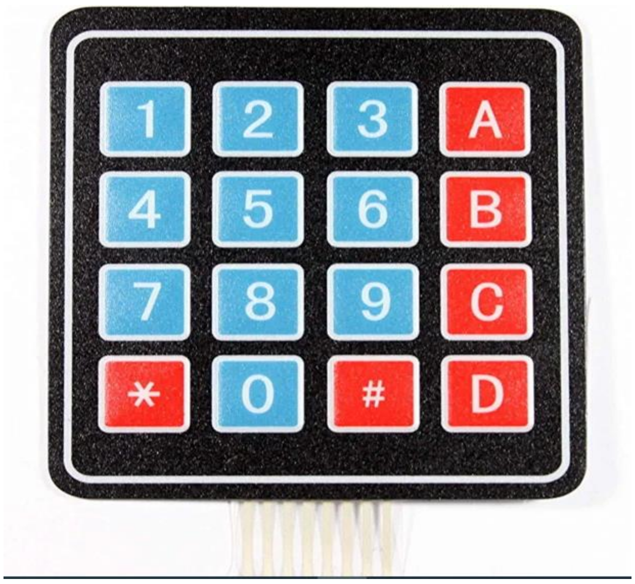
Figure 2 shows the Keypad

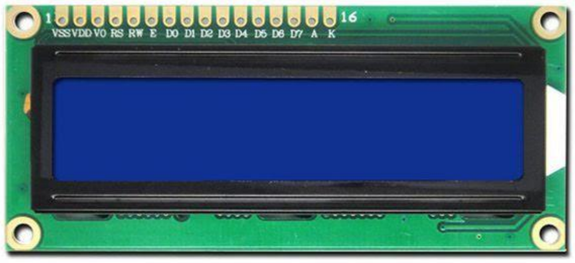
Figure 3 shows the LCD

Algorithm

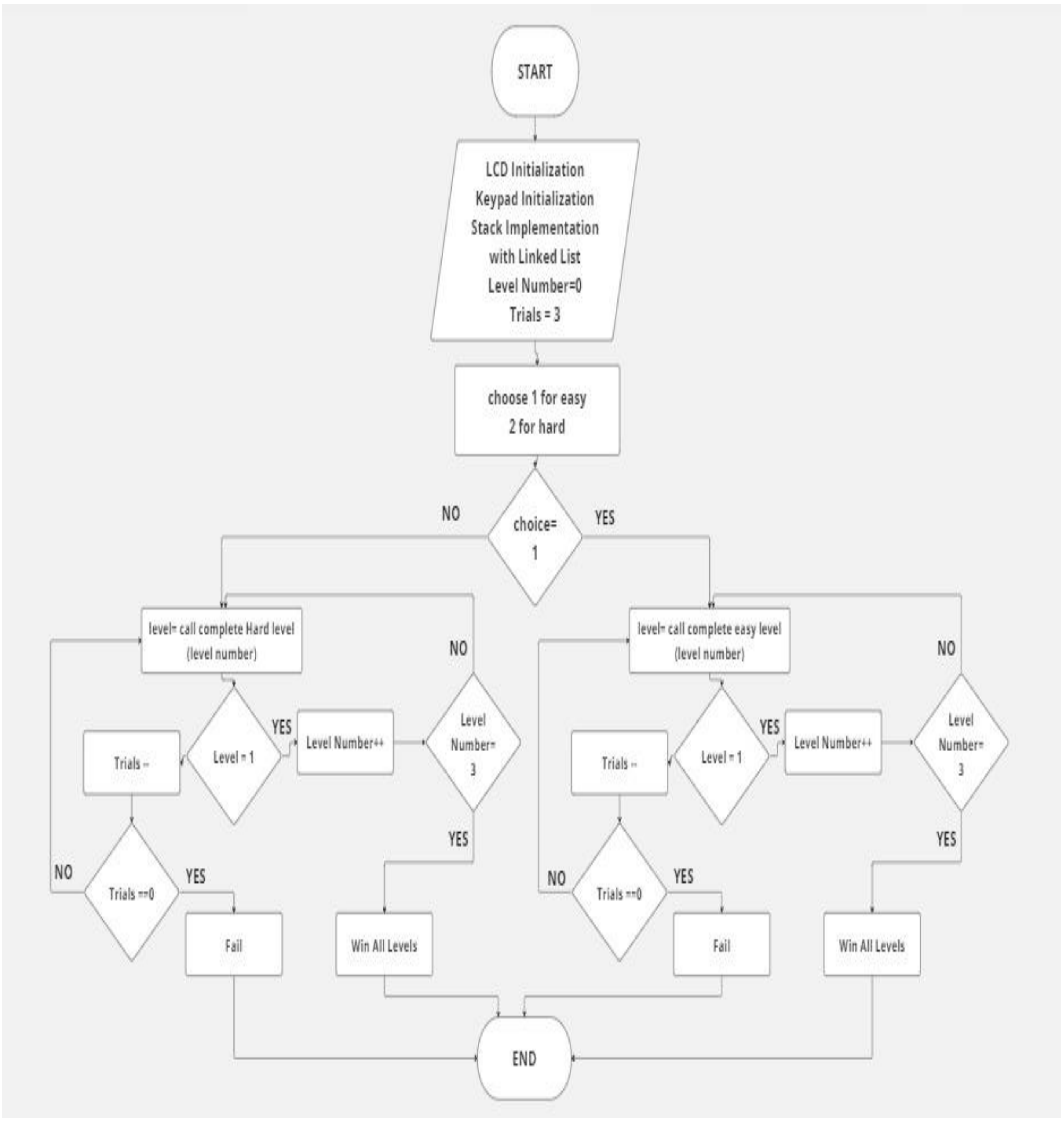

Figure 4 shows the Flow chart

Code

1- Implemented Stack:

- ✓ **Stack.h**

```
#ifndef Stack_h
#define Stack_h
#include <Arduino.h>
struct node{
    char data;
    node * next;
}; typedef node* ptr;
class Stack{
    private:
        ptr top=NULL;
    public:
        bool isEmpty();
        void push(char ele);
        char pop();
};
#endif
```

- ✓ **Stack.cpp**

```
#include "Stack.h"

bool Stack::isEmpty(){
    return top==NULL?true:false;
}
void Stack::push(char ele){
    ptr p=new node();
    p->data = ele;
    p->next=top;
    top=p;
}
char Stack::pop(){
    if (!isEmpty()){
        ptr temp = top;
        top=top->next;
        char val =temp->data;
        delete(temp);
        return val;
    }
}
```

2- Main Code:

```
#include "Stack.h"
#include <LiquidCrystal.h>
#include<time.h>
#include <Keypad.h>
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
 const byte rows = 4; //four rows
  const byte cols = 4; //three columns
  char keys[rows][cols] = {
    {')',']','}', '/'},
    {'4','5','6', '*'},
    {'1','2','9', '-'},
    {'#','0','=', '+'}
  };
  byte rowPins[rows] = {A0,A1,A2,A3}; //connect to the row pinouts of the keypad
  byte colPins[cols] = {13, 10,9,8}; //connect to the column pinouts of the keypad
  Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );
char arr_symbols[]={'(','[','{','*','+','/','-','#','='};
int sizes[]={4,6,8};
int speeds[]={1500,1000,500};
Stack symbols;
void setLevel(int num_level){
    while(!symbols.isEmpty()){
        symbols.pop();
    }
    srand(time(0));
    for(int i=0;i<=sizes[num_level]-1;i++){
      int index=random(0,9);
      symbols.push(arr_symbols[index]);
      lcd.print(arr_symbols[index]);
    }
}
bool flagupdown;
int count_fixed=0;
bool flag_easy;
char upDown(int level_num,int count){
  unsigned long tup;
  unsigned long tblink;
  flagupdown=false;
  char key=keypad.getKey();
  bool flag=false;
```

```
    while(!flag){
          /// still there
      if (count_fixed+count>15){
        lcd.clear();
        count_fixed=0;
        count=0;
      }
       for (int i=count_fixed+count;i<=15;i++){
         for(int j=0;j<=1;j++){        // up and down
      tup=millis();
      bool flagblink=false;
      int level_speed = speeds[level_num];
      while( !flag && millis()-tup<level_speed){    // time for up only or down only
        lcd.setCursor(i, int(flagupdown));
        tblink=millis();
        while(!flag && millis()-tblink<130){  // time for blinking only
          key=keypad.getKey();
          if (key){
            flag=true;
          }
            //blink
          if (!flagblink){lcd.cursor();}
          else{lcd.noCursor();}
        }
        flagblink=!flagblink;
      }
      flagupdown=!flagupdown;
      if (flag || flag_easy){
        break;
      }
    }
    if (flag || flag_easy){
        break;
      }
      count_fixed++;
    }
    Serial.print(flag_easy);
  }
  return key;
}
bool isPair(char ch1,char ch2){
    if (ch1 =='(' && ch2==')'){
        return true;
    }else if (ch1 =='[' && ch2==']'){
        return true;
    }else if (ch1 =='{' && ch2=='}'){
        return true;
    }else if (ch1==ch2){
        return true;
    }
    return false;
}
bool check_flag(char key){
    lcd.print(key);
    bool loose=false;
    if (flagupdown){
        // up
        char top = symbols.pop();
       // Serial.print(isPair(top,key));
        if (isPair(top,key)){
            return true;
        }else{
          loose=true;
        }
    }else{
        loose=true;
    }
    if (loose){
        delay(500);
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print("LOOSER -_-");
        delay(1000);
        lcd.clear();
        return false;
    }
}
bool complete_level(int level_num){
  count_fixed=sizes[level_num];
  setLevel(level_num);
  bool isTrue = true;
  int count=0;
  while(!symbols.isEmpty() and isTrue){
      char key=upDown(level_num,count);
      isTrue=check_flag (key);
      count++;
  }
  if (isTrue)
      return true;
  return false;
```

```
}
void setup() {
  randomSeed(5);
  // put your setup code here, to run once:
  lcd.begin(16, 2);
  // turn on the cursor:
  Serial.begin(9600);
  lcd.cursor();
  lcd.clear();
  lcd.setCursor(3,0);
  lcd.print("-Ouzo- ^_^");
  delay(2000);
  lcd.clear();
  lcd.print("1- easy. 2- hard");
  char choose=keypad.waitForKey();
  if (choose=='1'){
    flag_easy=true;
  }else{
    flag_easy=false;
  }
}
int level_num=0;
int trials =1;
void loop() {
  lcd.clear();
  lcd.setCursor(5, 0);
  lcd.print("Level ");
  lcd.print(level_num+1);
  delay(1000);
  lcd.clear();
  bool done=complete_level(level_num);
  if (done){
    lcd.clear();
    lcd.setCursor(3, 0) ;
    lcd.print("lucky! ^_^");
    delay(1000);
    lcd.clear();
    level_num++;
  }else{
    if (trials<=3){
      lcd.setCursor(5,0);
      lcd.print("Trial ");
      lcd.print(trials);
      delay(500);
      lcd.clear();
      trials++;
     }else{
      lcd.setCursor(0,0);
      lcd.print("-_-Game Over-_-");
      delay(2000);
      lcd.clear();
      trials=1;
      level_num=0;
        lcd.print("1- easy. 2- hard");
        char choose=keypad.waitForKey();
        if (choose=='1'){
          flag_easy=true;
        }else{
          flag_easy=false;
        }
     }
   }
   if (level_num==3){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("^_^ Winner ^_^");
      delay(10000);
      lcd.clear();
      trials=1;
      level_num=0;
      lcd.print("1- easy. 2- hard");
      char choose=keypad.waitForKey();
      if (choose=='1'){
        flag_easy=true;
      }else{
        flag_easy=false;
      }
    }
 }
```

Simulation

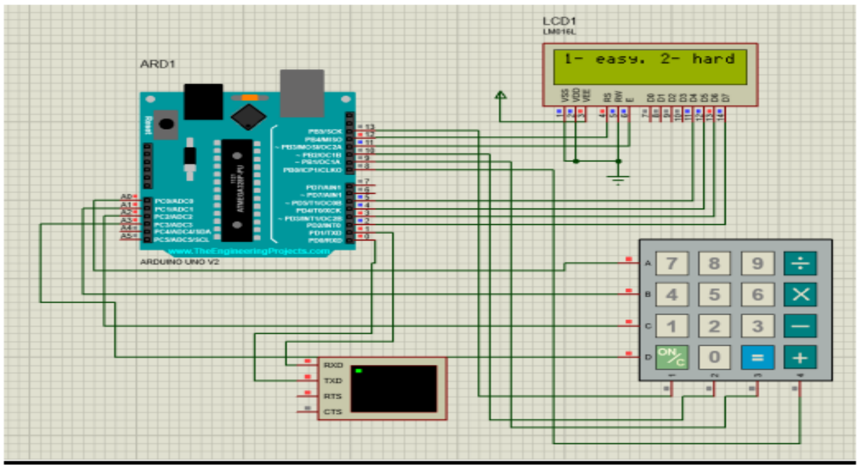

Figure 5 shows the beginning of OUZO

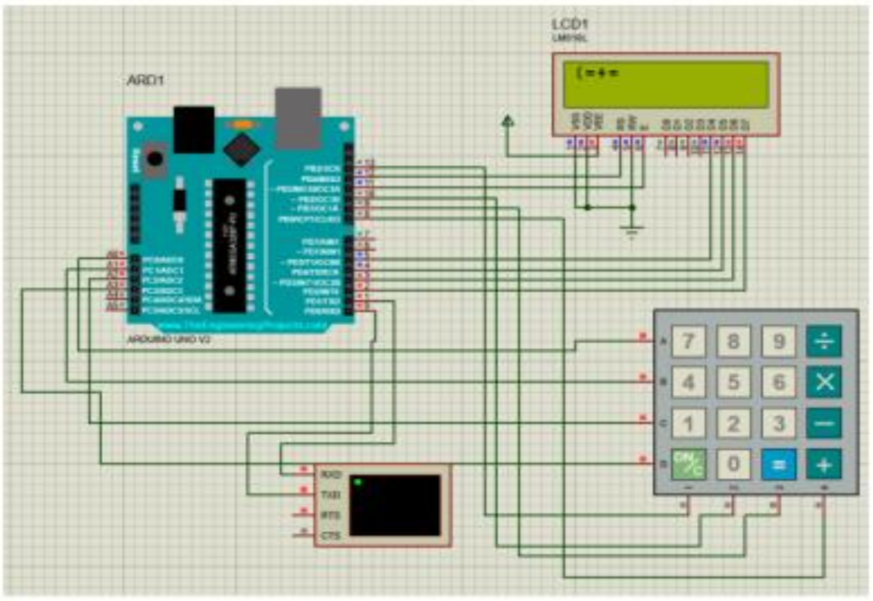

Figure 6 shows Level 1.

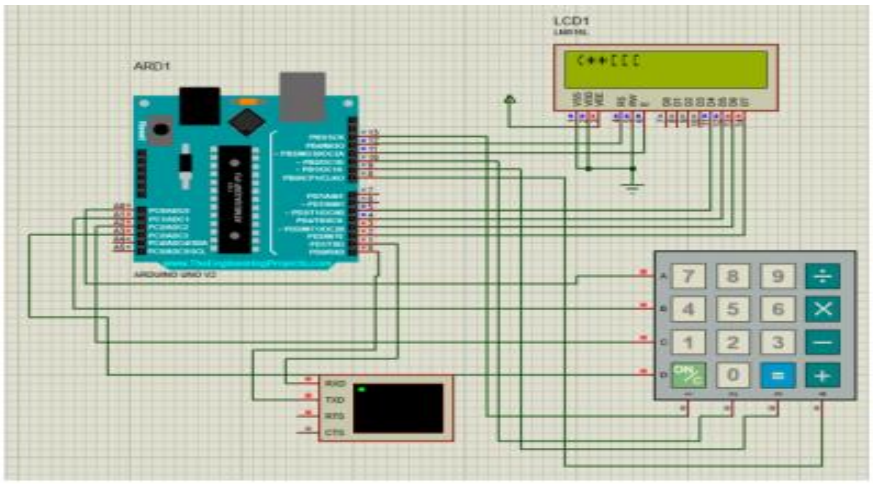

Figure 7 shows Level 2.

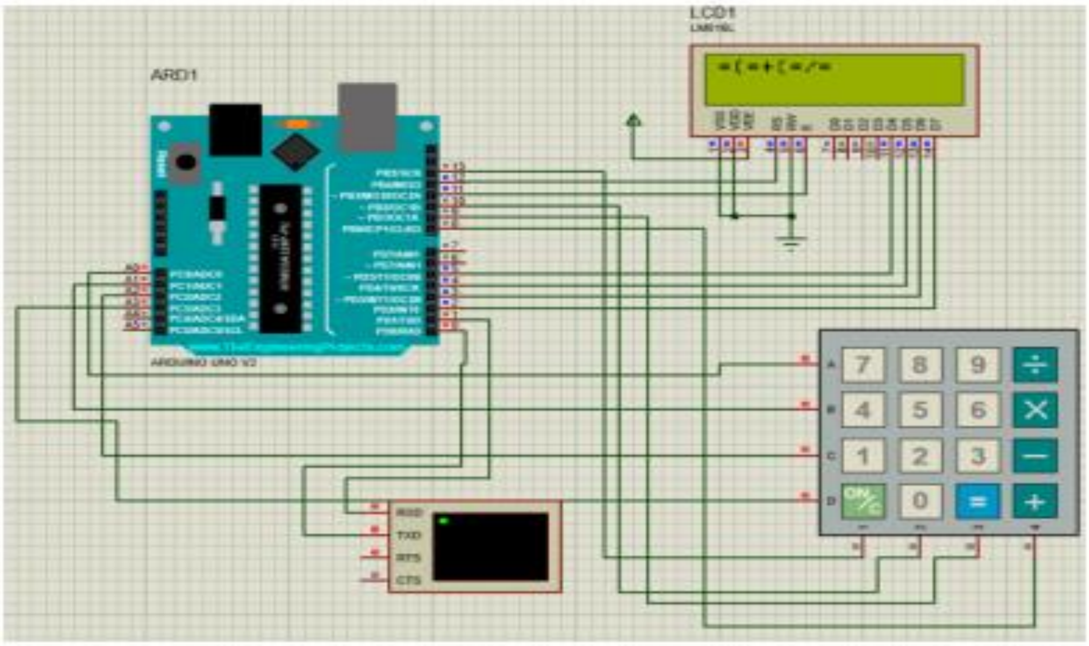

Figure 8 shows Level 3.

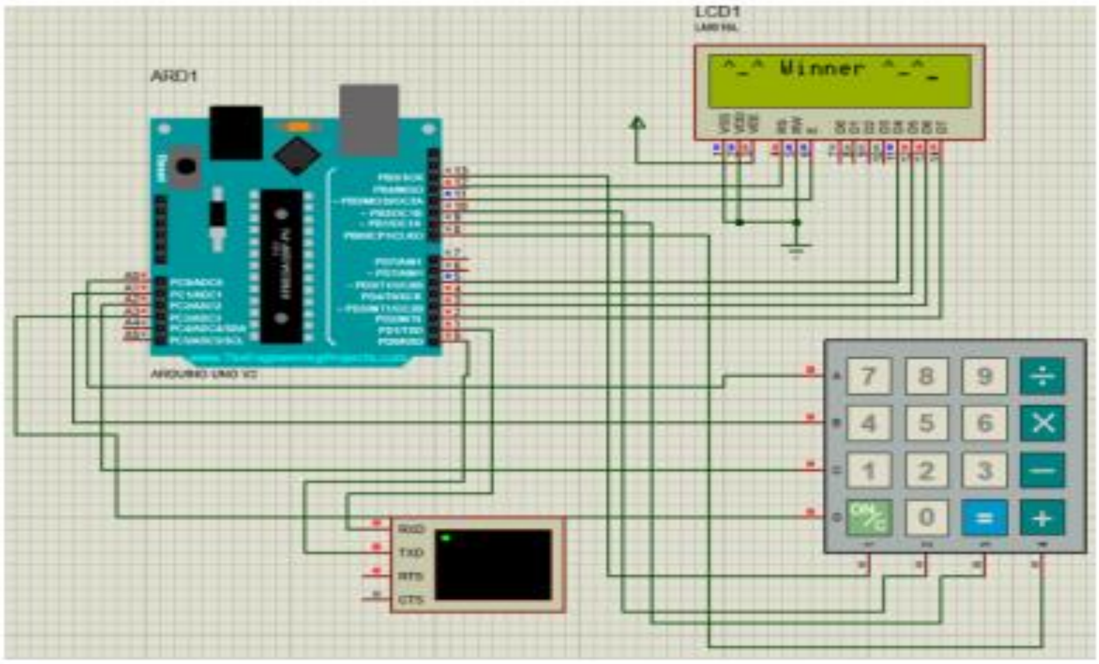

Figure 9 shows win.

Hardware Implementation

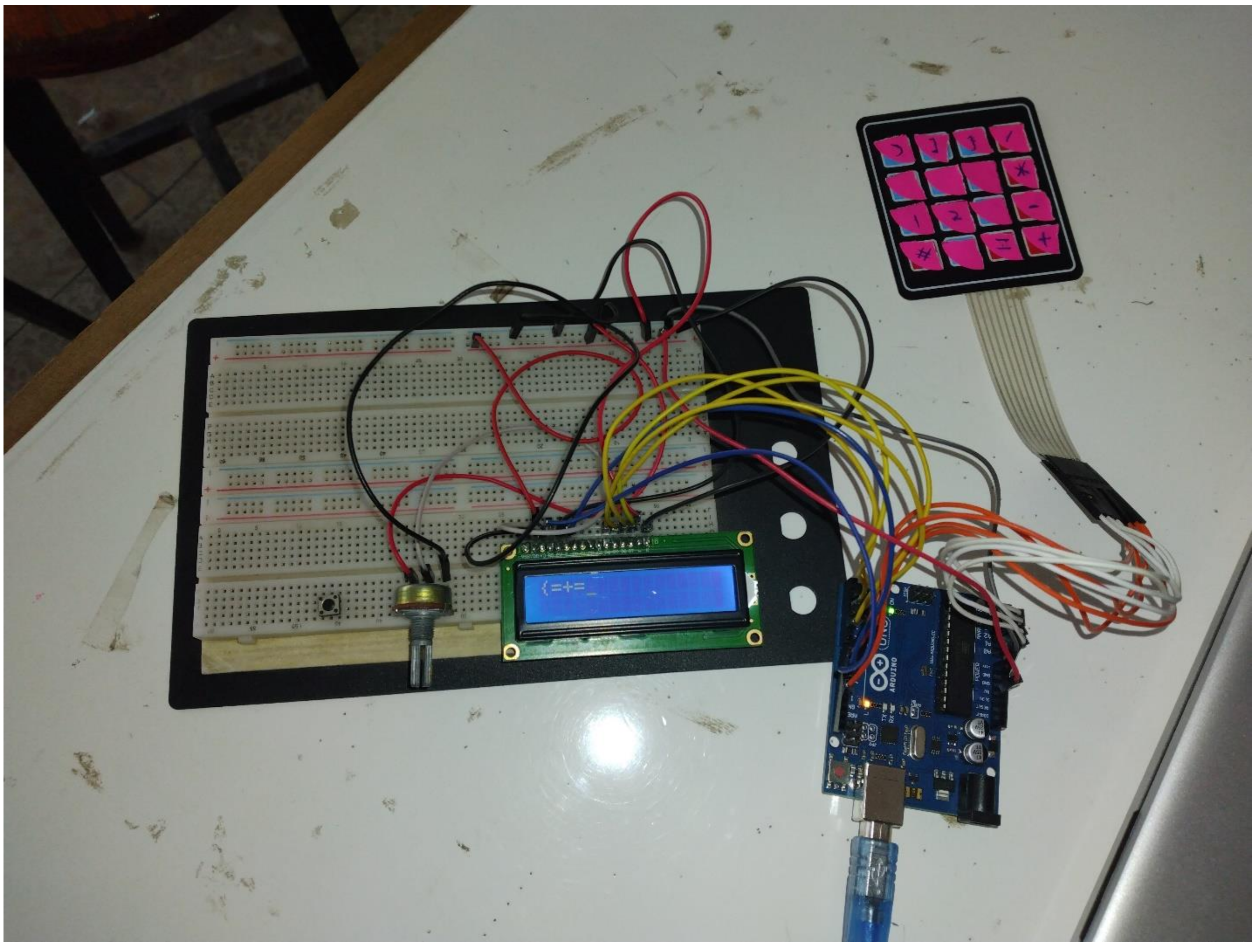

Figure 10 shows the Hardware

Conclusion

We take the stack implemented by the linked list as a connection between the data structure algorithms and OUZO game. The implementation saves the storage for the Arduino and saves the time for interesting playing. The user can easily choose between the two stages, easy or hard, and start playing. The user has three trials per stage to win the hole levels. The hard has a feature of moving curser which if the user could not enter the correct pattern in a specific time, the pattern will be smashed. So, the user must depend on his memory to complete the level.