ArduMaze

Tomáš Dvořák

# Poděkování

Chtěl bych poděkovat především vedoucímu práce Mgr. Lukáši Rýdlovi za veškeré rady a věnovaný čas, které práci pomohly do finální podoby. Také bych rád poděkoval vedení Gymnázia Matyáše Lercha za poskytnutí studijního prostředí na naší škole, včetně poskytnutí veškeré techniky a vybavení potřebných k vytvoření projektu. A nakonec bych rád poděkoval své rodině a přátelům, kteří mně podali zpětnou vazbu k uživatelské přívětivosti samotné hry, a především za jejich náležitou podporu.

Obsah

[Poděkování 2](#_Toc103854705)

[Úvod 4](#_Toc103854706)

[Teorie 5](#_Toc103854707)

[Hardware 5](#_Toc103854708)

[Nokia PCD 5440 5](#_Toc103854709)

[Joystick PS2 6](#_Toc103854710)

[Software 6](#_Toc103854711)

[Arduino IDE 6](#_Toc103854712)

[Code::Blocks 7](#_Toc103854713)

[Knihovny 7](#_Toc103854714)

[Implementace 8](#_Toc103854715)

[Sestavení Hardware 8](#_Toc103854716)

[MazeGenerator 9](#_Toc103854717)

[Vykreslování ve 3D 12](#_Toc103854718)

[Menu 14](#_Toc103854719)

[Závěr 16](#_Toc103854720)

[Zdroje 17](#_Toc103854721)

[Obrázky 17](#_Toc103854722)

# Úvod

Účelem této práce je prokázat, že i zdánlivě striktní pravidla mohou vést ke kreativní výzvě. Právě v informatice se tento jev velmi často objevuje na nejrůznějších příkladech. Ovšem žádný z nich není tak očividný jako problematika Arduina. Kvůli jeho zdánlivé jednoduchosti se může zdát, že se na této vývojové desce nedá mnoho uskutečnit. Ovšem pokud je člověk ochoten dát mu šanci, zjistí, že se mu nabízí opravdu unikátní výzvy. A jednu takovou jsem si vybral jako téma mé maturitní práce.

V rámci pokynů pro složení maturitní zkoušky jsem si pro demonstraci této výzvy zvolil za úkol naprogramovat 3D hru na vývojovou desku **Arduino Uno**. Mými limity jsou především **slabý výpočetní výkon**, **omezené možnosti periferií** a **nutnost programování v jazyce C**. Proto bylo důležité promyslet, jakým způsobem 3D hru naprogramuji bez možnosti použití složitých vývojových rozhraní (jako Unreal engine nebo Unity) či alespoň 3D knihoven, které se stejně pro Arduino nepoužívají.

Kvůli zvolenému hardwaru se značně omezily možnosti výběru hry, která by co možná nejvíce otestovala moje možnosti jakožto programátora a zároveň Arduina jakožto zdánlivě jednoduché učební pomůcky. Také jsem chtěl zvolit takovou hru, která by krásně demonstrovala a zároveň testovala uživatelovu prostorovou orientaci. A proto jsem se, po důkladné konzultaci s vedoucím práce, rozhodl pro vytvoření **náhodně generovaného labyrintu**, který by se hrál z **pohledu první osoby a** **imitoval prostorovou hloubku**.

Následně jsem si dal za úkol celou hru udělat uživatelsky přívětivou a to tak, že jí „zabalím“ do **úvodního menu**, které bude fungovat jako domovská obrazovka hned po spuštění

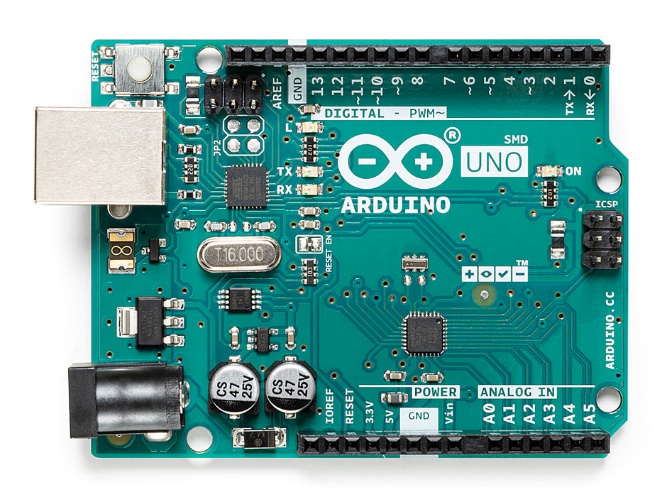
Nakonec tedy vznikla myšlenka *ArduMaze*, tedy 3D labyrintové hry z první osoby vytvořené bez jakékoliv reálné prostorové reprezentace. Jejím cílem je dosáhnout funkční, hratelné a prezentace schopné verze.

# Teorie

## Hardware

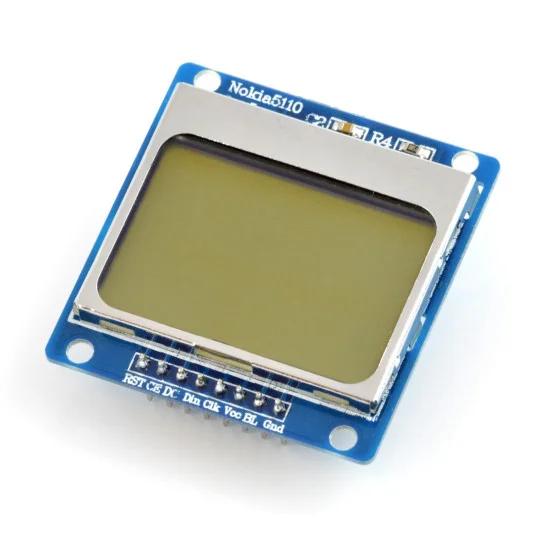
#### Arduino Uno

Jedná se o mikrokontrolerovou desku založenou na čipech *ATMega328*. Má 14 digitálních vstupních/výstupních pinů, 6 analogových vstupních pinů, 16MHz keramický rezonátor, USB převodník, power jack, ICSP konektor a tlačítko „reset“.

Napájení je řešeno přes připojení USB k počítači, jehož alternativou je i adaptér/baterie skrze zmíněný power jack, ale pro mé účely stačí pouze připojení k PC přes USB, které stejně využívám k nahrávání kódu (případně k napájení přes USB power banku). Vstupní napětí je tedy 5V (přes USB).

Obrázek : Arduino Uno Rev3 SMD

### Nokia PCD 5440

Populární LCD displej z populárního telefonu. Rozlišení 84x48 pixelů dnes není nic zázračného, ovšem znamená to, že je kompaktní a nenáročný na obsazenou pameť, což je pro můj projekt žádoucí. Důležité je zmínit především 8 pinů, přes které je displej připojený k Arduinu. Jsou to reset (RST) – restart LCD, chip enable (CE) – zapnutí LCD, data/command (DC) – přepínač posílání dat nebo příkazů, data input (DIN) – datová linka, clock (CLK) – hodinový signál, common collector voltage (VCC) – 5V napájení, backlight (BL) – napájení podsvícení, ground (GND) – uzemění, v tomto pořadí.

Obrázek : Nokia LCD 5110

### Joystick PS2

Analogový Joystick s dvěma 10K potenciometry umožňující snímání pohybu v ose X a Y, funguje jako napěťový dělič v rozsahu od 0V do 5V. Zároveň obsahuje tlačítko. Nalezneme na něm piny GND, +5V, VRX, VRY a SW, v tomto pořadí. Piny VRX a VRY jsou výstupem příslušného napěťového děliče v osách x a y.

Obrázek : Joystick PS2

## Software

### Arduino IDE

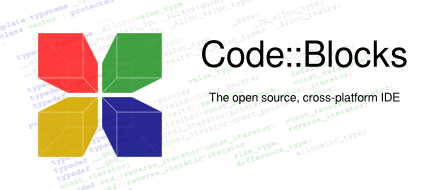
Veškeré programovaní Arduina se odehrává v open-source vývojovém prostředí Arduino IDE. Tato aplikace je dostupná pro Windows, Mac OS X i Linux a podporuje jazyky C a C++. Zatímco knihovny jsou obvykle psány objektově v C++, samotné projekty je zvykem psát v čistém C.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 4: Arduino IDE

### Code::Blocks

Podobně jako Arduino IDE je i Code::Blocks volně dostupné, open-source vývojové prostředí pro C/C++. Jedná se o univerzálnější a obsáhlejší vývojové prostředí. Můj projekt by se bez něj obešel, ovšem například psaní prvních prototypů kódu v něm bylo pohodlnější, především kvůli vestavěnému terminálu, který umožňuje rychlou kontrolu jeho výstupu).

Obrázek : Code::Blocks

### Knihovny

V informatice se jedná o soubor funkcí a procedur, který může být sdílen více počítačovými programy. Účelem knihovny je usnadnění tvorby zdrojového kódu tím, že programátor může využít již vytvořené kódy v jiných programech.

#### U8G2

U8g2 je monochromatická grafická knihovna pro vestavěná zařízení. U8g2 podporuje 3 různé vykreslovací metody: *Full screen buffer*, *Page mode* a *U8x8 mode*.



Obrázek : U8G2 Logo

ArduMaze plně používá metodu Full screen buffer, která je nejrychlejší, umožňuje využití všech vykreslovacích funkcí, každopádně je nejnáročnější na paměť RAM. U8x8 je nevhodný kvůli neschopnosti vykreslovat cokoliv, kromě textu. Page mode sice nepoužívá paměť RAM, oproti metodě Full screen buffer, zato používá paměť samotného displeje, což vede k překreslování jeho obsahu pomocí tzv. stránkování. Problém by nastal při rychlém překreslování displeje, kdy by se projevil efekt stránkování jako pravidelné vymazání a následné vykreslení scény, což je u mého projektu nežádoucí

# Implementace

Celý projekt bylo nejprve důležité si rozdělit do jednotlivých úkolů, které jsou snadno splnitelné a navazují na sebe tak, že dohromady vedou k finální verzi projektu. Této fázi projektu se jinak říká *dekompozice problému.* Tvorbu ArduMaze jsem si rozdělil následovně:

1. Sestavit Hardware
2. Naprogramovat MazeGenerator
3. Naprogramovat 3D vykreslování
4. Vytvořit Menu

V případě Menu není dané, že musí být hotové v závislosti na předchozích krocích (což dokazuje fakt, že to byla první hotová část kódu). Každopádně zbytek úkolů je jasný a chronologický. Dále je potřeba i tyto úkoly více analyzovat.

## Sestavení Hardware

Periferie je k Arduinu připojená pomocí drátků k nepájivému poli. Displej je připojen k digitálním pinům následovně: (CLK → pin 13, DIN → pin 11, CE → pin 10, DC → pin 9, RST → pin 8). Páčka Joysticku je připojena k analogovým pinům (VRX → pin A0, VRY → pin A1) a tlačítko k digitálnímu pinu (SW → pin 7). Samotné Arduino je pak potřeba mít připojené přes USB k počítači či k adaptéru.

Obsah obrázku text, elektronika, obvod

Popis byl vytvořen automaticky

Obrázek : Schéma ArduMaze

## MazeGenerator

Labyrint je rozdělený na jednotlivé buňky, které jsou reprezentované ve dvourozměrném poli jen jako číselné konstanty WALL (−1) – zeď, EMPTY (−2) – prázdno, PILLAR (−3) – pilíř, EXIT (−4) – východ z bludiště, MAP (−5) – políčko s bonusem mapa.

#define MAXX 15

#define MAXY 15 //předem definovaná velikost labyrintu

#define WALL -1

#define EMPTY -2

#define PILLAR -3

#define EXIT -4

#define MAP -5

char maze[MAXX][MAXY]; //dvourozměrné pole

Pro snadnou manipulaci s jejich souřadnicemi jsem vytvořil datový typ *struct cell*, který má parametry int x a int y. Pomocí něho se dá zapsat jakákoliv kombinace souřadnic a skrze ně nahlédnout do labyrintu (do pole) a zjistit tak hodnotu dané buňky (WALL, EMPTY,…).

struct cell { //datový typ struct

 char x;

 char y;

};

Poté je potřeba vyplnit labyrint zdmi a prázdnými buňkami.

for (int x = 0; x < MAXX; x++) { //vytvoří všechny 4 řady vnějších zdí

   maze[x][0] = WALL;

   maze[x][MAXY - 1] = WALL;

   maze[0][x] = WALL;

   maze[MAXX - 1][x] = WALL;

}

A aby to byl opravdový labyrint, je potřeba vytvořit prostory uvnitř. K tomu jsem použil metodu[[1]](#footnote-1) pilířů, které se vygenerují vždy na sudých polích (kromě vnějších zdí) a následně se staví zeď od náhodně vybraného pilíře v náhodném směru až ke zdi, čímž se zajistí, že po jejich vzájemném spojení zůstane vždy průchozí chodba.

const char plrsmax = ((MAXX - 3) / 2) \* ((MAXY - 3) / 2); //maximální počet možných pilířu při daných rozměrech labyrintu

struct cell pillar[plrsmax]; //pole pro uložení budoucích pilířů

int idx; //index jako ukazatel na pilíře v poli

for (int x = 2; x < MAXX - 1; x += 2) {

  for (int y = 2; y < MAXY - 1; y += 2) {

    maze[x][y] = PILLAR; //zaplní labyrint/zapíše buňky jako pilíř

    pillar[idx].x = x; //uloží vytvořené pilíře do pole

    pillar[idx].y = y;

    idx++;

  }

}

for (int plrs = plrsmax; plrs > 0; plrs--) {

  int idx = random(plrs); //zvolí se náhodný pilíř

  int dirplrs = random(4); //zvolí se náhodný směr

  if (dirplrs == 0) { //nahoru

    for (int y = pillar[idx].y; maze[pillar[idx].x][y] != WALL; y--) {

      maze[pillar[idx].x][y] = WALL;

    } //spojí pilíře se zdmi

  }

  if (dirplrs == 1)  { //doprava

    for (int x = pillar[idx].x; maze[x][pillar[idx].y] != WALL; x++) {

      maze[x][pillar[idx].y] = WALL;

    } //spojí pilíře se zdmi

  }

  if (dirplrs == 2)  { //dolů

    for (int y = pillar[idx].y; maze[pillar[idx].x][y] != WALL; y++) {

      maze[pillar[idx].x][y] = WALL;

    } //spojí pilíře se zdmi

  }

  if (dirplrs == 3) {                  //doleva

    for (int x = pillar[idx].x; maze[x][pillar[idx].y] != WALL; x--) {

      maze[x][pillar[idx].y] = WALL;

    } //spojí pilíře se zdmi

  }

  pillar[idx] = pillar[plrs - 1]; //na konci každého cyklu uloží poslední pilíř v poli na místo zvoleného pilíře, aby se následně mohlo toto pole zmenšit o jedna (inkrement je „plrs—„) a tím smazat poslední pilíř

Poté už stačí jen náhodně zapsat polohu startu, cíle a mapy.

do { //najde a uloží mapu do prázdné buňky

   mapPos.x = rand() % MAXX;

   mapPos.y = rand() % MAXY;

} while (maze[mapPos.x][mapPos.y] != EMPTY);

maze[mapPos.x][mapPos.y] = MAP; //vytvoří mapu v labyrintu

startPosX = 2 \* (rand() % (MAXX - 1) / 2) + 1; //náhodný start na sudých polích, aby se vyhnul pilířům

maze[startPosX][MAXY - 1] = EMPTY;    //vytvoří start v labyrintu

do {

 exitPosX = 2 \* (rand() % (MAXX - 1) / 2) + 1; // náhodný cíl na sudých polích, aby se vyhnul pilířům

}while (exitPosX == startPosX); //zajistí, aby start a cíl nebyli naproti sobě

maze[exitPosX][0] = EXIT; //vytvoří cíl v labyrintu

## Vykreslování ve 3D

Konečně se dostáváme k hlavnímu problému celého projektu. Ovšem není zas tak složitý, jak se může zdát. Stačí si totiž uvědomit, že pokud budeme hrát celou dobu z první osoby, uvidíme celou scénu „pod stejným úhlem“. Tudíž se nám zredukuje počet možných pohledů a nám tedy stačí (při dohledu čtyři buňky dopředu a tři do boku) zkontrolovat vždy hodnotu 11 buněk. Ovšem je potřeba si uvědomit, že jejich pozice se bude měnit v závislosti na orientaci hráče. Pro tento účel je tu funkce *getViewedCell()*, pomocí které jsem schopný zjistit pomocí relativní pozice buňky vůči orientaci hráče absolutní pozici buňky v labyrintu a zjistit tak rovnou i její hodnotu. Tímto způsobem se dá zkontrolovat všech 11 buněk a vykreslit tak výslednou scénu.

#define NORTH 0

#define EAST 1

#define SOUTH 2

#define WEST 3

char view = NORTH;   //začíná na hodnotě 0, tedy severu (v ovládání hráče je zajištěna cykličnost)

char getViewedCell(char x, char y) {

 char a, b;

 switch (view) {

   case NORTH:

     a = x;

     b = (-y);

     break;

   case EAST:

     a = y;

     b = x;

     break;

   case SOUTH:

     a = (-x);

     b = y;

     break;

   case WEST:

     a = (-y);

     b = (-x);

     break;

 }

 if (playerPos.x + a > MAXX || playerPos.x + a < 0 || playerPos.y + b > MAXY || playerPos.y + b < 0) {

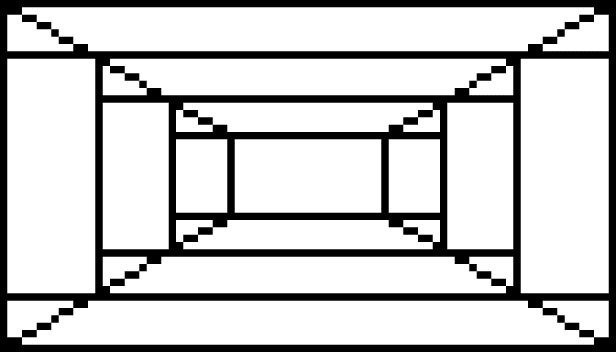
   return EMPTY; //všechny buňky mimo labyrint jsou prázdné

 } else {

   return maze[playerPos.x + a][playerPos.y + b]; //vrátí absolutní pozici v labyrintu

 }

}

Samotná funkce *draw3D()* má dvě fáze. Nejprve vykreslí mřížku pomocí funkce *drawGrid()*, která nakreslí rozvržení scény, ale bez jediné zdi.

Obsah obrázku šipka

Popis byl vytvořen automatickyTy se pak kreslí pomocí předem popsané funkce a to tak, že se každá zkontroluje a nakreslí zvlášť a postupuje se od nejvzdálenější buňky po nejbližší. Tím je docíleno zakrytí zdí zdmi před nimi.

void draw3D() {   //Toto je jen část celé funkce

**u8g2**.clearBuffer();´ //smaže obsah bufferu (vymaže obrazovku)

 drawGrid();

 if (getViewedCell(0, 3) == WALL && getViewedCell(0, 3) != EXIT) {

**u8g2**.drawBox(32, 18, 20, 12);

 }

 if (getViewedCell(-1, 3) == WALL) {

**u8g2**.drawBox(23, 18, 9, 12);

 }

 if (getViewedCell(1, 3) == WALL) {

**u8g2**.drawBox(52, 18, 9, 12);

 }

**u8g2**.sendBuffer(); //pošle obsah bufferu (vykreslí aktuální stav obrazovky)

}

## Menu

Základní myšlenka Menu je mít nějaký kód, který poběží od začátku, a uvítá hráče do hry. Arduino totiž nedokáže „ukončit kód“. Běží, dokud je připojené ke zdroji.

Koncept je rozdělen do dvou částí: funkce, která bude přijímat vstup a promítat ho do menu ve formě změny pozice kurzoru, a funkce, která bude dané menu (včetně kurzoru) vykreslovat.

Funkce *doMenu()* tedy kontroluje pohyb joysticku nahoru a dolů a poté volá funkci *drawMenu()*.

char menuPosition = 0; //pozice kurzoru

const char \*menuItems[] = //jednotlivé položky menu

{

 "Start Game",

 "Options",

 "Help",

};

char doMenu() {

 waitForNoInput(); //čeká se, dokud se nedetekuje žádný input

 do {

   int input = read\_joystick(); //read\_joystick() převádí analogový input na digitální hodnoty (UP, DOWN, LEFT, RIGHT a SELECT)

   switch (input) {

     case DOWN:

       menuPosition++;

       break;

     case UP:

       menuPosition--;

       break;

     case SELECT:

       return menuPosition; //vrátí aktuální pozici kurzoru

   }

   if (menuPosition > 2) {

     menuPosition = 2;

   }

   if (menuPosition < 0) { //obě podmínky řeší necykličnost menu

     menuPosition = 0;

   }

   drawMenu();

   delay(200); //čeká se 200ms (pro klidnější pohyb kurzoru)

   waitForAnyInput(); //čeká se, dokud se nedetekuje nějaký input

 } while (1); //opakuje se, dokud „pravda“ (tedy pořád)

}

Ta už si vykreslí titulek a jednotlivé položky (menu items) a postará se, aby byly na střed.

void drawMenu() {

 setFontDefault(); //funkce s výchozími hodnotami pro psaní fontů

**u8g2**.clearBuffer();

**u8g2**.drawLine(0, 0, **u8g2**.getDisplayWidth(), 0); //nakreslí úsečku

**u8g2**.setCursor((**u8g2**.getDisplayWidth() - **u8g2**.getStrWidth("ArduMaze XXX")) / 2, ROW - 3);

**u8g2**.println("ArduMaze ");

**u8g2**.println(versionArduMaze);

**u8g2**.drawLine(0, ROW - 2, **u8g2**.getDisplayWidth(), ROW - 2);

 for (byte m = 0; m < 3; m++)  {

   if (m == menuPosition) { //nakreslí kurzor, pokud je na dané položce

**u8g2**.drawRFrame(0, (m + 1) \* ROW , **u8g2**.getDisplayWidth(), ROW, 3);

   }

**u8g2**.setCursor((**u8g2**.getDisplayWidth()

**u8g2**.getStrWidth(menuItems[m])) / 2, (m + 1) \* ROW + 9);

**u8g2**.println( menuItems[m]); //vykreslí všechny položky

 }

**u8g2**.sendBuffer();

}

Stejným způsobem jsou řešená i ostatní menu (pause menu, exit menu…).

# Závěr

Co se původně zdálo jako malá výzva, se nakonec ukázalo jako unikátní a zajímavý projekt. Během něho jsem se lépe seznámil s fungováním Arduina, jak po hardwarové, tak po softwarové stránce. Ze začátku bylo obtížné si dobře rozvrhnout práci, aby byla seřazená od jednodušší po složitější úkoly. Proto jsem se také ze začátku vrhnul na tvorbu menu, na kterém jsem si osvojil práci s knihovnou U8G2. Poté už byla každá grafická funkce maličkostí. Samotný MazeGenerator byl zase dobrou zkouškou znalostí základů jazyka C a datových typů.

Celý projekt měl ze začátku pozvolné tempo, ale postupně jsem získával větší a větší motivaci. Obzvlášť ve chvíli, kdy má člověk hotový první prototyp, se kterým si může hrát a v reálnem čase upravovat, se celá práce stane více osobní a nabere tempo. Ovšem zde nastává další problém. V rámci euforie pro fungující prototyp se začnou objevovat nové nápady na zlepšení, které vůbec nebyly v původním plánu. Nemám na mysli chyby či drobné optimalizace, které jsou nezbytné udělat, pokud se na ně přijde, ale jde o úplně nové funkce či mechaniky, které by celý projekt posunuly, ale trvaly by dlouho, než by se realizovaly. V ten moment je důležité nenechat se utopit v těchto nápadech a nechat si je bokem, dokud se nevyřeší první funkční verze. I já jsem se málem stal obětí tohoto fenoménu, a i když už teď mám v plánu dodělat animace, nastavení velikosti labyrintu za běhu, přehlednější zobrazení mapy a exitu v labyrintu, musel jsem je pro účely maturitní zkoušky odložit. Nemluvě o „krabičce“, do které by se celá „konzole“ dala zavřít a měla by tak kompaktní a praktický tvar (ve kterém by nepřekážely drátky).

Celkově však tato výzva naplnila má očekávání a naučila mě spoustu cenných dovedností. Sice nevím, kdy v budoucnu mě čeká další podobná práce, ale pokud ano, budu na ní díky tomuto projektu připraven.

# Zdroje

* Arduino Uno Rev3 SMD. Store.arduino.cc [online]. [cit. 2022-05-08]. Dostupné z: <https://store.arduino.cc/products/arduino-uno-rev3-smd>
* LCD displej Nokia 5110. Dratek.cz [online]. [cit. 2022-05-08]. Dostupné z: <https://dratek.cz/arduino/878-arduino-lcd-display-nokia-5110-modre-podsviceni.html>
* Joystick PS2. Dratek.cz [online]. [cit. 2022-05-08]. Dostupné z: <https://dratek.cz/arduino/884-joystick-ps2.html>
* Arduino IDE [online]. [cit. 2022-05-08]. Dostupné z: <https://www.javatpoint.com/arduino-ide>
* Knihovna. It-slovnik.cz [online]. [cit. 2022-05-08]. Dostupné z: <https://it-slovnik.cz/pojem/knihovna>
* U8G2. Github.com: olikraus [online]. [cit. 2022-05-08]. Dostupné z: <https://github.com/olikraus/u8g2/wiki>
* Mircosoft. Lekce 2 – Generování náhodného bludiště [online]. [cit. 2022-05-08]. Dostupné z: <https://www.itnetwork.cz/algoritmy/bludiste/algoritmus-tvorba-nahodneho-bludiste>

# Obrázky

1. Arduino Uno Rev3 SMD. Cdn.shopify.com [online]. [cit. 2022-05-08]. Dostupné z: <https://cdn.shopify.com/s/files/1/0438/4735/2471/products/A000073_00.front_804x603.jpg?v=1637829596>
2. Arduino Uno Rev3 SMD. *Cdn3.botland.store* [online]. [cit. 2022-05-08]. Dostupné z: <https://cdn3.botland.store/59174-large_default/lcd-graphic-display-84x48px-nokia-5110-blue.jpg>
3. Nokia LCD 5110. *Dratek.cz* [online]. [cit. 2022-05-08]. Dostupné z: <https://dratek.cz/photos/produkty/d/0/884.jpg?m=1502871177>
4. Code::Blocks. Javatpoint.com [online]. [cit. 2022-05-08]. Dostupné z: <https://www.codeblocks.org/docs/cb_splash.png>
5. U8G2 Logo. Raw.githubusercontent.com [online]. [cit. 2022-05-08]. Dostupné z: [https://raw.githubusercontent.com/wiki/olikraus/u8g2/img/  
   uc1701\_dogs102\_uno\_board\_320.jpg](https://raw.githubusercontent.com/wiki/olikraus/u8g2/img/uc1701_dogs102_uno_board_320.jpg)

1. Více na webové stránce <https://www.itnetwork.cz/algoritmy/bludiste/algoritmus-tvorba-nahodneho-bludiste>. [↑](#footnote-ref-1)