MINISTERUL EDUCAŢIEI ŞI CERCETĂRII ŞTIINŢIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

Proiect Arduino

# Proiectare cu microprocesoare

**SnakeGame**

Moldovan Paul Andrei
30235

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

# Contents

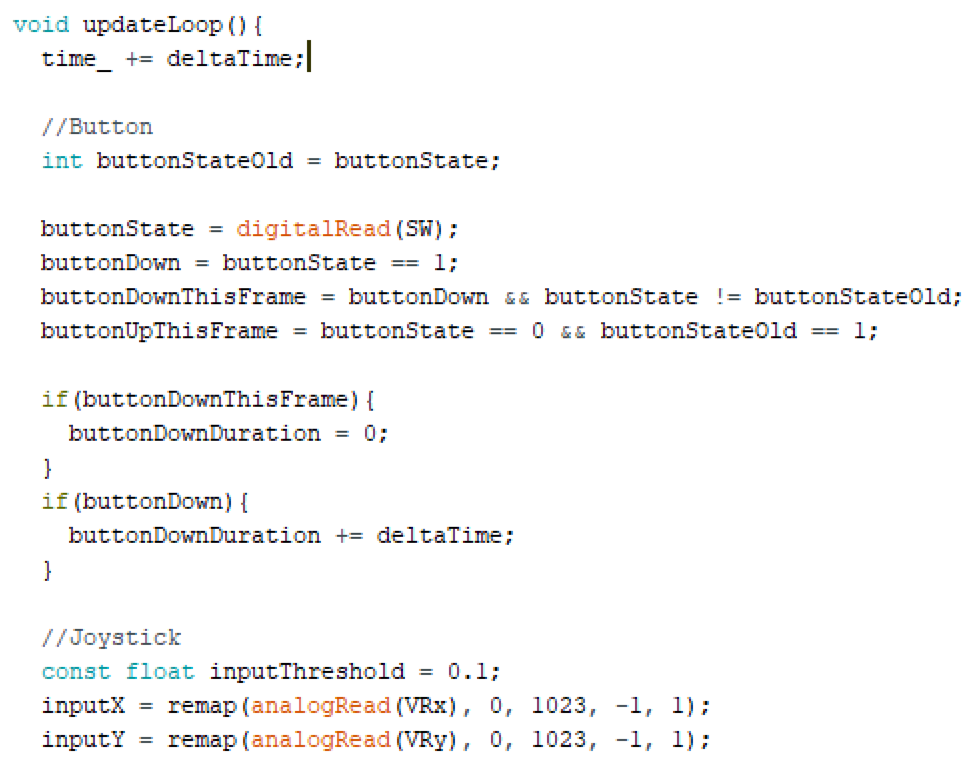
# 1. Prezentarea temei

Tema proiectului este un joc clasic de snake. Pentru reprezentarea jocului am hotarat sa folosesc doua afisoare matrice de leduri de 8x8 iar pentru control un joystick.

## 2. Solutia

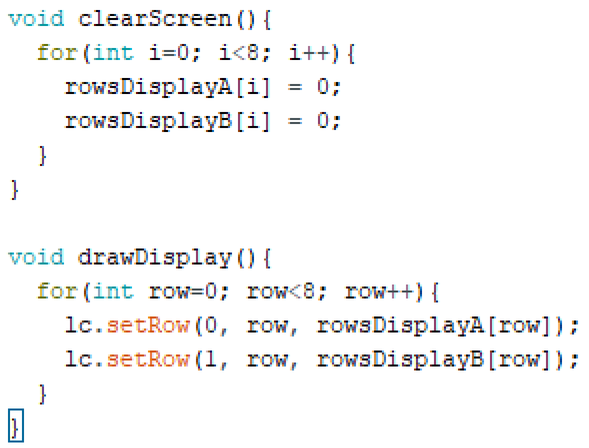
Pentru solutia proiectului am decis sa folosesc libraria LedControl care ofera o solutie usoara de a controla o matrice de leduri prin functiile pe care le ofera.

Am definit o functie pentru citirea controalelor de la joystick.



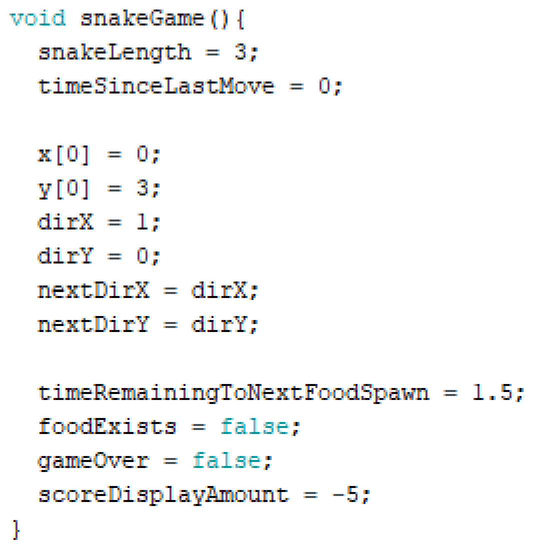
```
void updateLoop(){
  time_ += deltaTime;

  //Button
  int buttonStateOld = buttonState;

  buttonState = digitalRead(SW);
  buttonDown = buttonState == 1;
  buttonDownThisFrame = buttonDown && buttonState != buttonStateOld;
  buttonUpThisFrame = buttonState == 0 && buttonStateOld == 1;

  if(buttonDownThisFrame){
    buttonDownDuration = 0;
  }
  if(buttonDown){
    buttonDownDuration += deltaTime;
  }

  //Joystick
  const float inputThreshold = 0.1;
  inputX = remap(analogRead(VRx), 0, 1023, -1, 1);
  inputY = remap(analogRead(VRy), 0, 1023, -1, 1);
```

Pentru controlul matricelor de leduri am ales sa declar doua matrici si valorile din ele sa le arat pe leduri. Am declarat functii pentru a aprinde si opri ledurile de pe matrice.



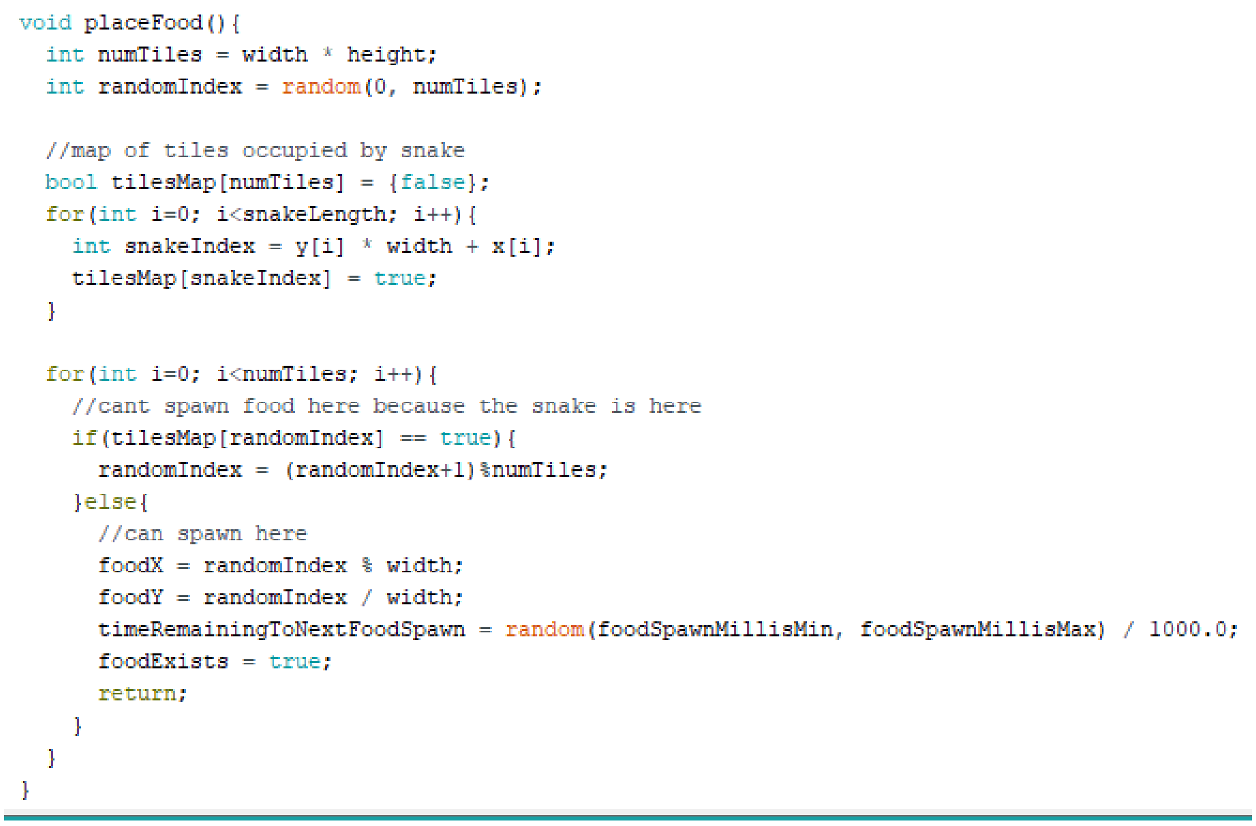
```
void clearScreen(){
  for(int i=0; i<8; i++){
    rowsDisplayA[i] = 0;
    rowsDisplayB[i] = 0;
  }
}

void drawDisplay(){
  for(int row=0; row<8; row++){
    lc.setRow(0, row, rowsDisplayA[row]);
    lc.setRow(1, row, rowsDisplayB[row]);
  }
}
```

Pentru partea de joc trei functii. Una se ocupa de initializarea jocului.



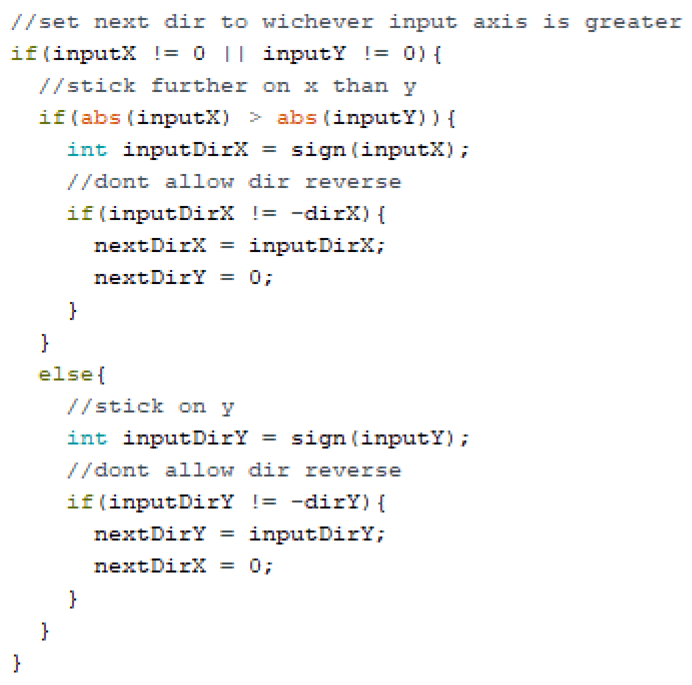
```
void snakeGame(){
  snakeLength = 3;
  timeSinceLastMove = 0;

  x[0] = 0;
  y[0] = 3;
  dirX = 1;
  dirY = 0;
  nextDirX = dirX;
  nextDirY = dirY;

  timeRemainingToNextFoodSpawn = 1.5;
  foodExists = false;
  gameOver = false;
  scoreDisplayAmount = -5;
}
```

O functie care se ocupa de locul unde sa apara punctele de mancare pentru snake, placeFood(). Verifica pozitia curenta a snake-ului pentru a ca punctul de mancare sa apara pe snake. Pozitia este generata random.



```
void placeFood(){
  int numTiles = width * height;
  int randomIndex = random(0, numTiles);

  //map of tiles occupied by snake
  bool tilesMap[numTiles] = {false};
  for(int i=0; i<snakeLength; i++){
    int snakeIndex = y[i] * width + x[i];
    tilesMap[snakeIndex] = true;
  }

  for(int i=0; i<numTiles; i++){
    //cant spawn food here because the snake is here
    if(tilesMap[randomIndex] == true){
      randomIndex = (randomIndex+1)%numTiles;
    }else{
      //can spawn here
      foodX = randomIndex % width;
      foodY = randomIndex / width;
      timeRemainingToNextFoodSpawn = random(foodSpawnMillisMin, foodSpawnMillisMax) / 1000.0;
      foodExists = true;
      return;
    }
  }
}
```

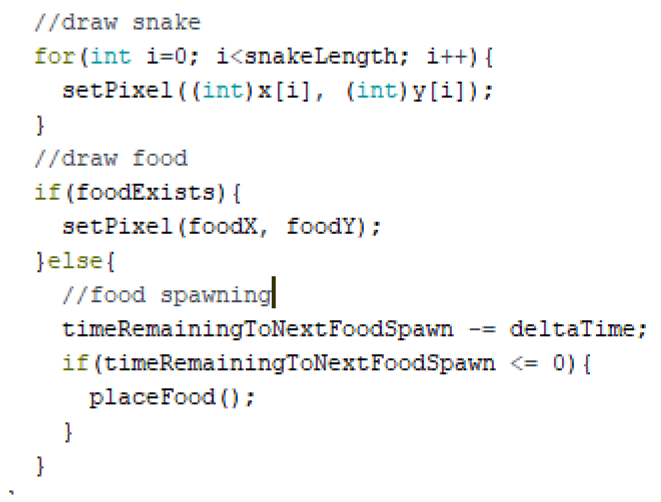
Si o functie care se ocupa de logica jocului, updateSnakeGame(). In aceasta se verifica daca jocul s-a terminat, daca da atunci reincepe de la inceput. Daca jocul ruleaza atunci verifica controlul primit de la joystick pentru a face urmatoarea miscare.



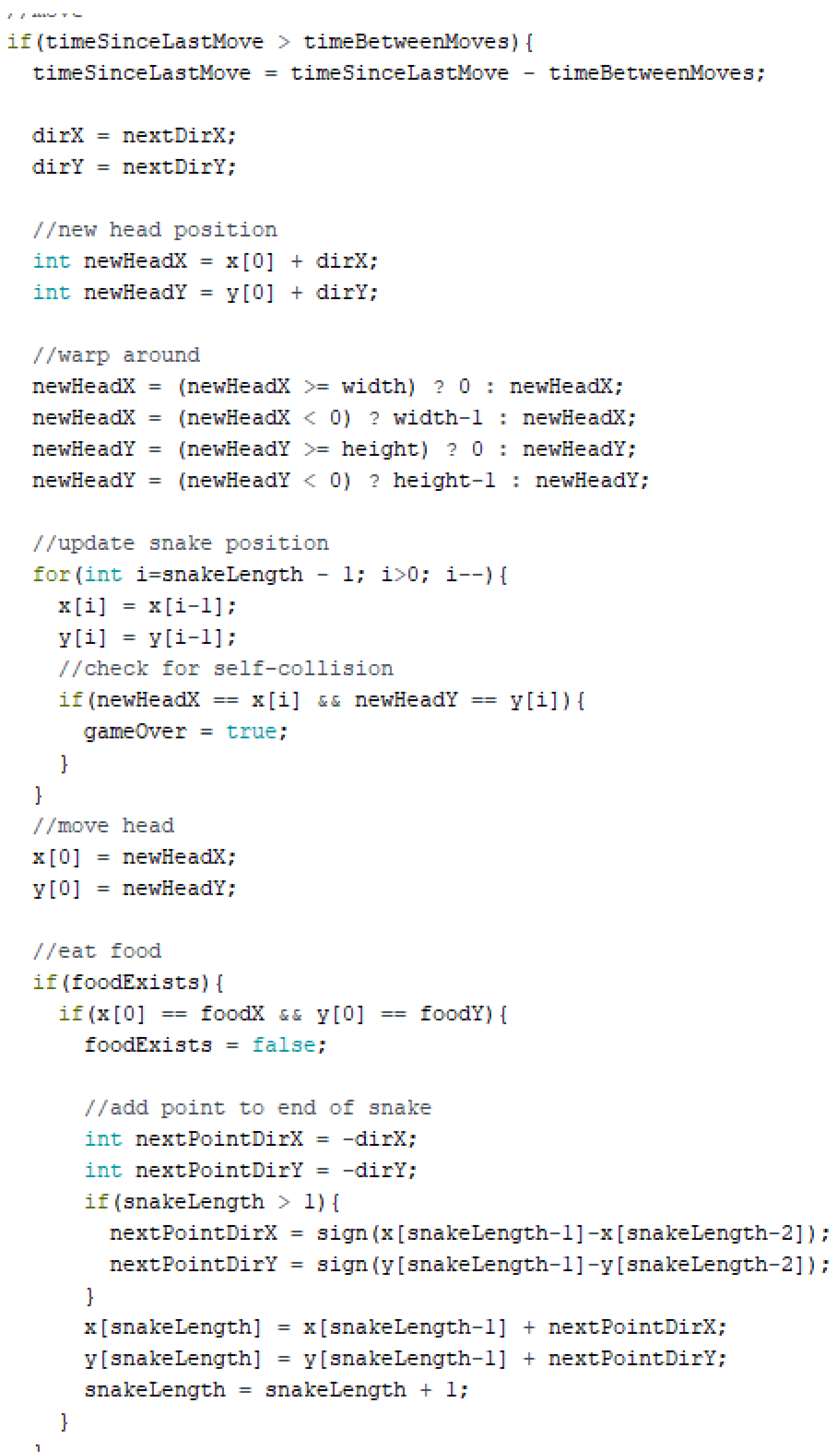
```
//set next dir to wichever input axis is greater
if(inputX != 0 || inputY != 0){
  //stick further on x than y
  if(abs(inputX) > abs(inputY)){
    int inputDirX = sign(inputX);
    //dont allow dir reverse
    if(inputDirX != -dirX){
      nextDirX = inputDirX;
      nextDirY = 0;
    }
  }
  else{
    //stick on y
    int inputDirY = sign(inputY);
    //dont allow dir reverse
    if(inputDirY != -dirY){
      nextDirY = inputDirY;
      nextDirX = 0;
    }
  }
}
```

Dupa aceea este prezentata logica de miscare si ce se intampla cand mananca un punct de mancare.

Si in final sunt afisate pe leduri snake-ul si punctele de mancare.

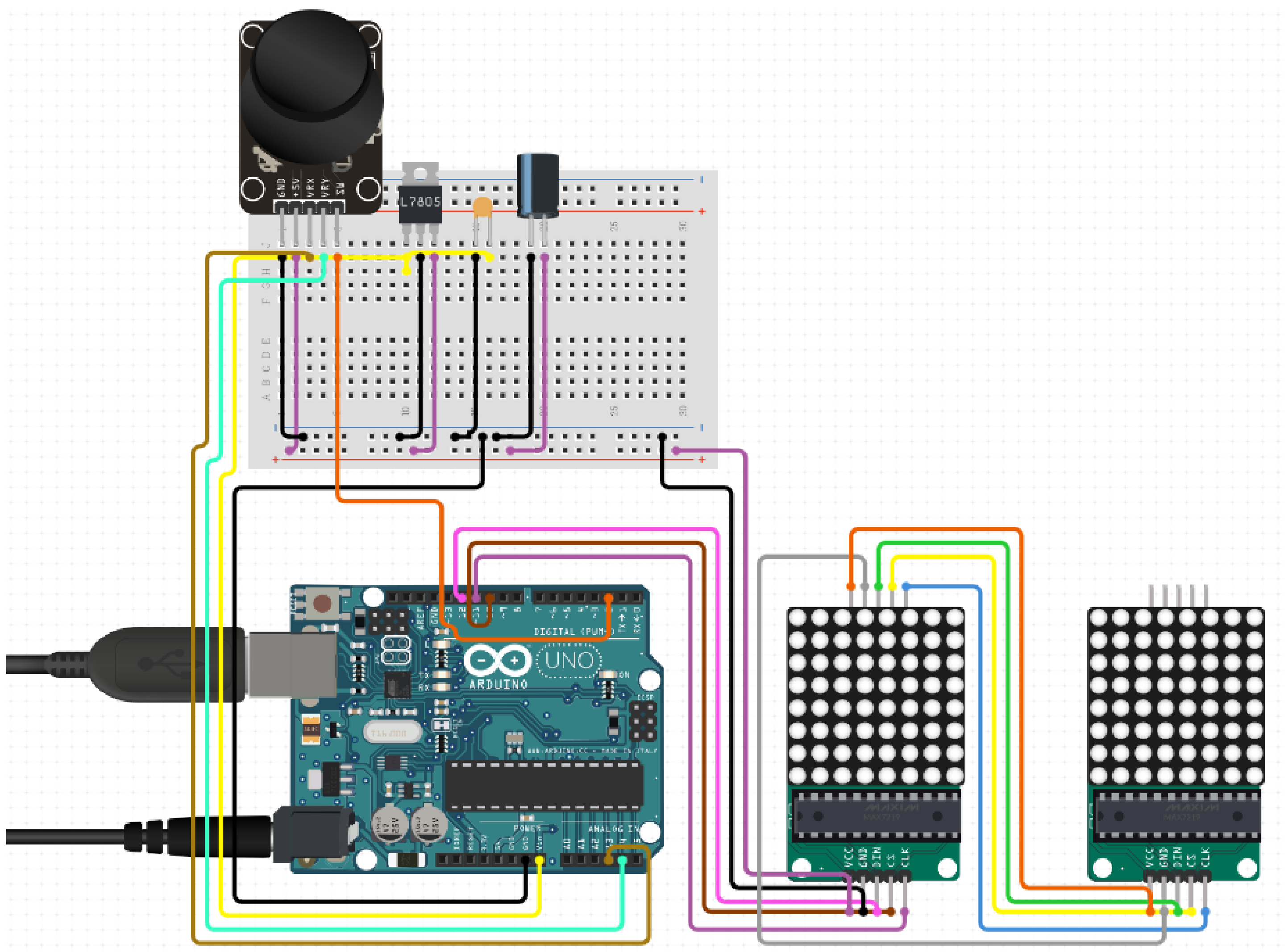


```
//draw snake
for(int i=0; i<snakeLength; i++){
  setPixel((int)x[i], (int)y[i]);
}
//draw food
if(foodExists){
  setPixel(foodX, foodY);
}else{
  //food spawning
  timeRemainingToNextFoodSpawn -= deltaTime;
  if(timeRemainingToNextFoodSpawn <= 0){
    placeFood();
  }
}
```



```
//move
if(timeSinceLastMove > timeBetweenMoves){
  timeSinceLastMove = timeSinceLastMove - timeBetweenMoves;

  dirX = nextDirX;
  dirY = nextDirY;

  //new head position
  int newHeadX = x[0] + dirX;
  int newHeadY = y[0] + dirY;

  //warp around
  newHeadX = (newHeadX >= width) ? 0 : newHeadX;
  newHeadX = (newHeadX < 0) ? width-1 : newHeadX;
  newHeadY = (newHeadY >= height) ? 0 : newHeadY;
  newHeadY = (newHeadY < 0) ? height-1 : newHeadY;

  //update snake position
  for(int i=snakeLength - 1; i>0; i--){
    x[i] = x[i-1];
    y[i] = y[i-1];
    //check for self-collision
    if(newHeadX == x[i] && newHeadY == y[i]){
      gameOver = true;
    }
  }
  //move head
  x[0] = newHeadX;
  y[0] = newHeadY;

  //eat food
  if(foodExists){
    if(x[0] == foodX && y[0] == foodY){
      foodExists = false;

      //add point to end of snake
      int nextPointDirX = -dirX;
      int nextPointDirY = -dirY;
      if(snakeLength > 1){
        nextPointDirX = sign(x[snakeLength-1]-x[snakeLength-2]);
        nextPointDirY = sign(y[snakeLength-1]-y[snakeLength-2]);
      }
      x[snakeLength] = x[snakeLength-1] + nextPointDirX;
      y[snakeLength] = y[snakeLength-1] + nextPointDirY;
      snakeLength = snakeLength + 1;
    }
  }
```

## 3. Diagrama circuit



# 4. Specificatii

## Hardware:

- Arduino Uno
- Led Matrix 2x
- Joystick
- Conectori
- Breadboard
- USB A-B

## Software:

- ArduinoIDE
- LedControl library

# 5. Referinte

https://www.arduino.cc/en/Main/Software

https://www.arduino.cc/en/Guide/Libraries#toc2

https://ardushop.ro/ro/

http://www.wayoda.org/arduino/ledcontrol/

https://github.com/SebLague/Monster-Console