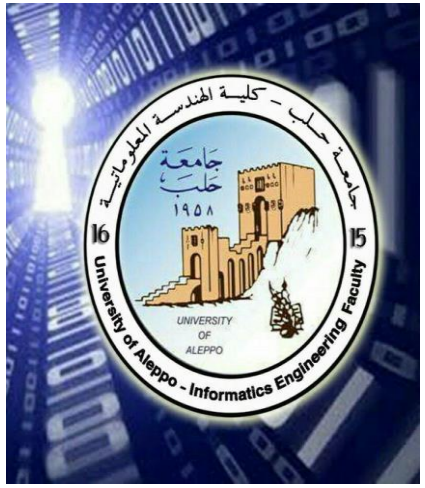


مشروع Snake with arduino

الطلاب المشاركين :

١-محمد نعسان نجار

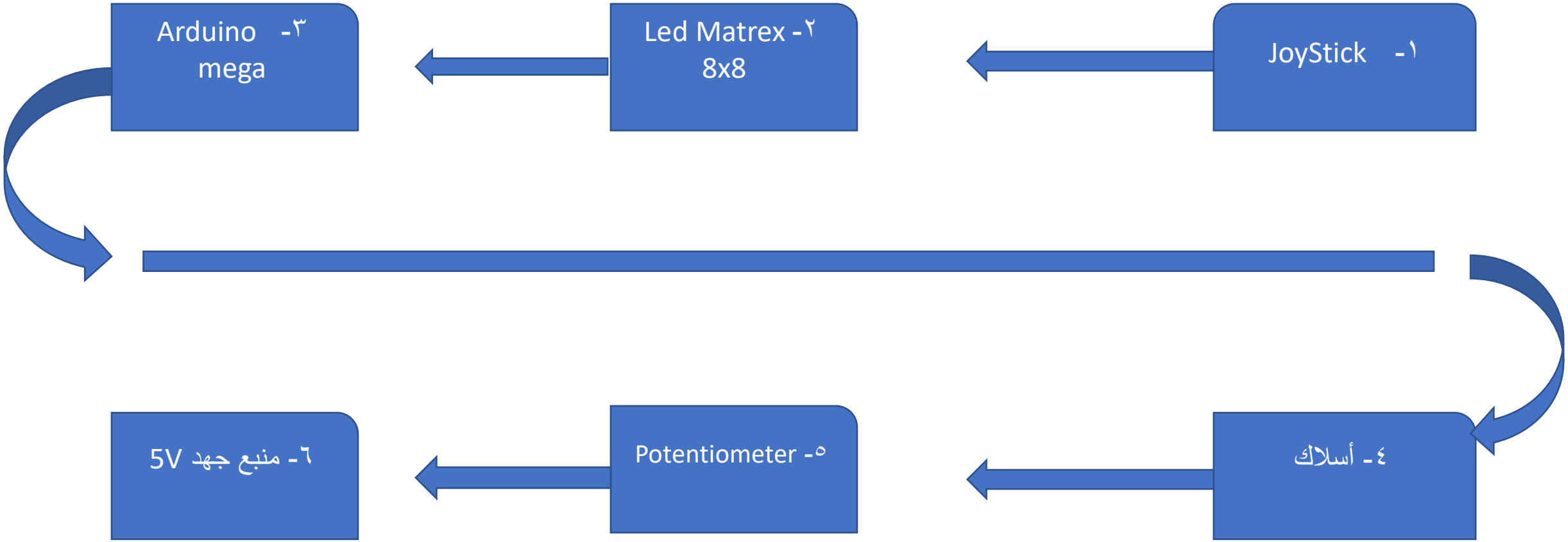
٢-محمد علي نجار

٣-خالد أحمد الخلف

بإشراف الدكتور : عبد القادر غزال

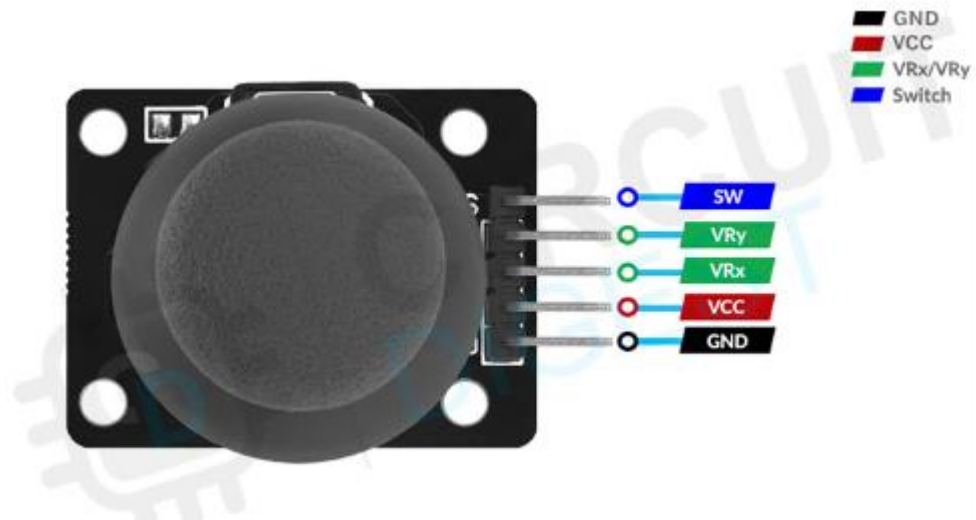
بتاريخ : ٢٠٢٢/١٢/٠٧

المكونات :


١- JoyStick
٢- Led Matrex 8x8
٣- Arduino mega
٤- أسلاك
٥- Potentiometer
٦- منبع جهد 5V

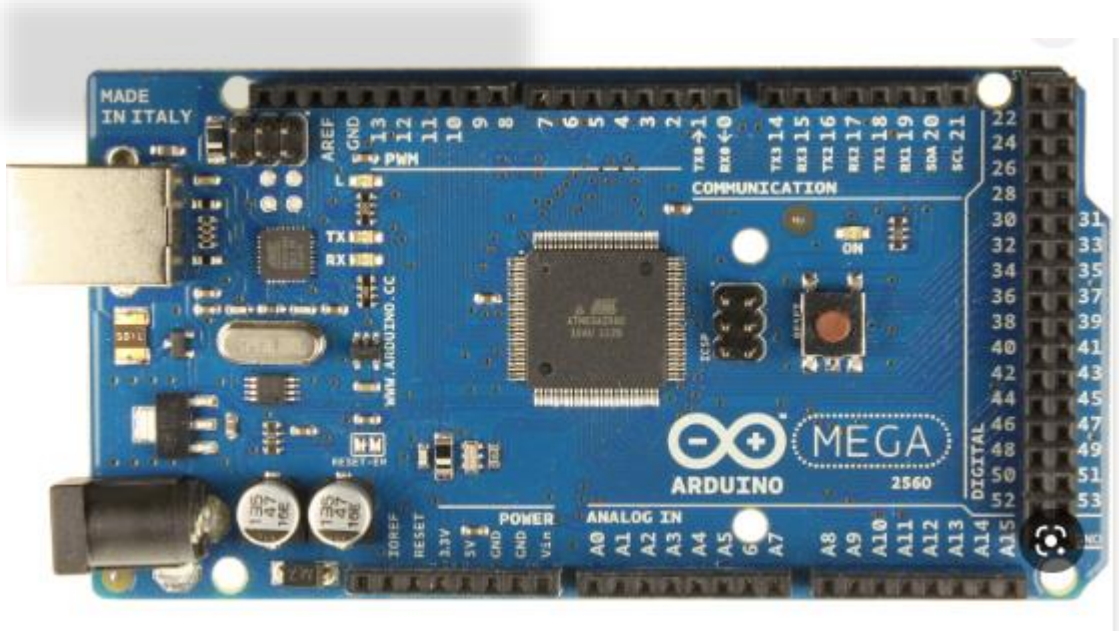
(عصا التحكم) Joystaick : هي جهاز ادخال يتكون من عصا تدور على قاعدة وتبلغ عن زاويتها أو اتجاهها للجهاز الذي يتحكم فيه. ذراع التحكم ، المعروف أيضًا باسم **عمود التحكم** ، هو جهاز التحكم الرئيسي .

غالبًا ما تستخدم عصي التحكم ، للتحكم في ألعاب الفيديو ، وعادة ما يكون بها زر ضغط واحد أو أكثر يمكن للكمبيوتر أيضًا قراءة حالتها. من الأشكال الشائعة لعصا التحكم المستخدمة في وحدات تحكم العاب الفيديو الحديثة العصا التناظرية



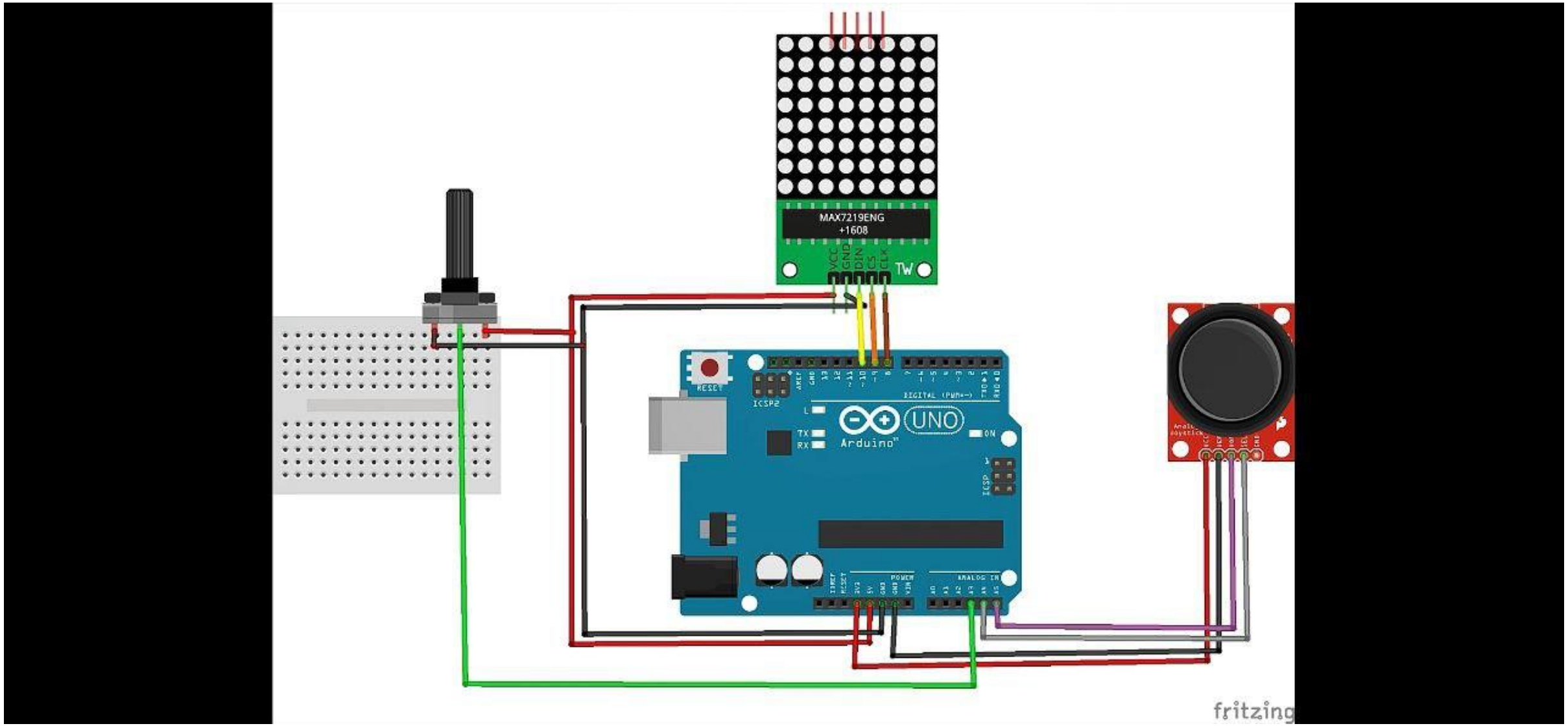
(مقاومة متغيرة) Potentiometer : هو مقاوم ثلاثي الأطراف مزود بوصلة انزلاقية أو دوارة تشكل مقسم جهد قابل للضبط . ذا تم استخدام طرفين فقط ، طرف واحد والممسحة ، فإنها تعمل **كمقاوم متغير** أو مقاومة **متغيرة**

تُستخدم مقاييس الجهد بشكل شائع للتحكم في الأجهزة الكهربائية مثل أدوات التحكم في مستوى الصوت في المعدات الصوتية. يمكن استخدام مقاييس الجهد التي يتم تشغيلها بواسطة آلية **كمحولات طاقة** ، على سبيل المثال ، في عصا التحكم.



Arduino mega : هي **لوحة متحكم تحتوي على ٥٤ دبوس إدخال / إخراج رقمي** (يمكن استخدام ١٥ منها كمخرجات تضمين عرض النبضة) ، و ١٦ مدخلًا تناظريًا ، و ٤ (منافذ تسلسلية للأجهزة) ، ومذبذب كريستالي ١٦ ميجا هرتز ، ووصلة (يو اس بى) ، ومقبس طاقة ، وزر إعادة الضبط.

مخطط المشروع :



خطوات تجهيز المشروع :

الادوات اللازمة :

١- JoyStick
٢- Led Matrex 8x8
٣- Arduino mega
٤- مقاومة متغيرة
٥- اسلاك
٦- منبع جهد 5V
٧- كود برمجي للمشروع
٨- كابل USP

اولاً : علينا وصل الأدوات بالأردوينو

١. **Led Matrex** : نقوم يتغذيتها VCC & GND والدبابيس DIN & CS & CLK نوصلهما بمخرخات الأردوينو حسب الموجود لدينا بالكود البرمجي

٢. **JoyStick :** كذلك نقوم بتغذية العصا VCC & GRN والدبابابيس VRX & VRY نوصلهما بالمخلات الموضحة بالكود

٣. **Potentiometer :** بعد تغذيتها موجب وسالب نقوم بوصل القطب الأوسط بالمدخل الموضح بالكود

ثانيا :

نقوم بوضع الكود البرمجي الخاص بنا على تطبيق أردوينو ومن ثم فحص الكود لخلوه من الأخطاء ثم نتأكد من من اختيار المنفذ بعدها نقوم برفع الكود على شريحة الأردوينو سيتم تفعيل الكود في حال كان الوصل صحيح

أهداف المشروع :

الهدف الرئيسي : الانطلاق من مشروع بسيط أولاً لكسب الثقة بالنفس وبعدها كتابة أفكار مطروحة على الإنترنت او أفكار لحل مشكلة تواجهنا او تبسيط عمل شيئ ما .

الأهداف العامة : في زمننا هذا اصبح الارتكاز على الألعاب شيئ مهم جداً وهذه اللعبة البسيطة جزء مصغر من الألعاب المسلية

الهدف الشخصي :

من خلاله تعرفنا على شريحة (الأردوينو)

تعرفنا على طريقة كتابة الأكواد البرمجية على الأردوينو

إمكانية تطوير المشروع :

يمكن تطوير المشروع بعدة طرق :

١- يمكن وضع مكبر صوت في حال الثعبان ذاد نقطة يصدر صوت

٢- زيادة شاشة led matrix 8x8 بشاشة مثلاً led matrix 12x8

٣- وضع الثعبان داخل Box وفي حال اصطدام الثعبان في الصندوق يؤدي للخسراة

الكود البرمجي :

```
#include "LedControl.h"
struct Pin {
  static const short joystickX = A2;
  static const short joystickY = A3;
  static const short joystickVCC = 15;
  static const short joystickGND = 14;
  static const short potentiometer = A5;
  static const short CLK = 10;
  static const short CS  = 11;
  static const short DIN = 12;
};
const short intensity = 6;
const short messageSpeed = 5;
const short initialSnakeLength = 3;
void setup() {
  Serial.begin(115200);
  initialize();
  calibrateJoystick();
  showSnakeMessage();
}
void loop() {
  generateFood();
  scanJoystick();
  calculateSnake();
  handleGameStates();
}
```

```
LedControl matrix(Pin::DIN, Pin::CLK, Pin::CS, 1);
struct Point {
  int row = 0, col = 0;
  Point(int row = 0, int col = 0): row(row), col(col)
{}
};
struct Coordinate {
  int x = 0, y = 0;
  Coordinate(int x = 0, int y = 0): x(x), y(y) {}
};
bool win = false;
bool gameOver = false;
Point snake;
Point food(-1, -1);
Coordinate joystickHome(500, 500);
int snakeLength = initialSnakeLength;
int snakeSpeed = 1;
int snakeDirection = 0;
const short up     = 1;
const short right  = 2;
const short down   = 3;
const short left   = 4;
const int joystickThreshold = 160;
const float logarithmity = 0.4;
int gameboard[8][8] = {};
void generateFood() {
  if (food.row == -1 || food.col == -1) {
    if (snakeLength >= 64) {
      win = true;
      return; }
    do {
      food.col = random(8);
      food.row = random(8);
    } while (gameboard[food.row][food.col] > 0);}}
```

```
void scanJoystick() {
  int previousDirection = snakeDirection;
  long timestamp = millis();

  while (millis() < timestamp + snakeSpeed) {
    float raw =
mapf(analogRead(Pin::potentiometer), 0, 1023, 0,
1);
    snakeSpeed = mapf(pow(raw, 3.5), 0, 1, 10,
1000);
    if (snakeSpeed == 0) snakeSpeed = 1;
    analogRead(Pin::joystickY) < joystickHome.y -
joystickThreshold ? snakeDirection = up    : 0;
    analogRead(Pin::joystickY) > joystickHome.y +
joystickThreshold ? snakeDirection = down  : 0;
    analogRead(Pin::joystickX) < joystickHome.x -
joystickThreshold ? snakeDirection = left  : 0;
    analogRead(Pin::joystickX) > joystickHome.x +
joystickThreshold ? snakeDirection = right : 0;
    snakeDirection + 2 == previousDirection &&
previousDirection != 0 ? snakeDirection =
previousDirection : 0;
    snakeDirection - 2 == previousDirection &&
previousDirection != 0 ? snakeDirection =
previousDirection : 0;
    matrix.setLed(0, food.row, food.col, millis() %
100 < 50 ? 1 : 0);}}
```

```
void calculateSnake() {
 switch (snakeDirection) {
  case up:
   snake.row--;
   fixEdge();
   matrix.setLed(0, snake.row, snake.col, 1);
   break;
  case right:
   snake.col++;
   fixEdge();
   matrix.setLed(0, snake.row, snake.col, 1);
   break;
  case down:
   snake.row++;
   fixEdge();
   matrix.setLed(0, snake.row, snake.col, 1);
   break;
  case left:
   snake.col--;
   fixEdge();
   matrix.setLed(0, snake.row, snake.col, 1);
   break;
  default:
   return;}
 if (gameboard[snake.row][snake.col] > 1 && snakeDirection != 0) {
  gameOver = true;
  return;}
 if (snake.row == food.row && snake.col == food.col) {
  food.row = -1;
  food.col = -1;
  snakeLength++;
  for (int row = 0; row < 8; row++) {
   for (int col = 0; col < 8; col++) {
    if (gameboard[row][col] > 0 ) {
     gameboard[row][col]++;}}}}
 gameboard[snake.row][snake.col] = snakeLength + 1;
 for (int row = 0; row < 8; row++) {
  for (int col = 0; col < 8; col++) {
   if (gameboard[row][col] > 0 ) {
    gameboard[row][col]--;}
   matrix.setLed(0, row, col, gameboard[row][col] == 0 ? 0 : 1);}}}
```

```
void fixEdge() {
 snake.col < 0 ? snake.col += 8 : 0;
 snake.col > 7 ? snake.col -= 8 : 0;
 snake.row < 0 ? snake.row += 8 : 0;
 snake.row > 7 ? snake.row -= 8 : 0;
}
void handleGameStates() {
 if (gameOver || win) {
  unrollSnake();
  showScoreMessage(snakeLength - initialSnakeLength);
  if (gameOver) showGameOverMessage();
  else if (win) showWinMessage();
  win = false;
  gameOver = false;
  snake.row = random(8);
  snake.col = random(8);
  food.row = -1;
  food.col = -1;
  snakeLength = initialSnakeLength;
  snakeDirection = 0;
  memset(gameboard, 0, sizeof(gameboard[0][0]) * 8 * 8);
  matrix.clearDisplay(0);}}
void unrollSnake() {
 matrix.setLed(0, food.row, food.col, 0);
 delay(800);
 for (int i = 0; i < 5; i++) {
  for (int row = 0; row < 8; row++) {
   for (int col = 0; col < 8; col++) {
    matrix.setLed(0, row, col, gameboard[row][col] == 0 ? 1 : 0);}}
    delay(20);
  for (int row = 0; row < 8; row++) {
   for (int col = 0; col < 8; col++) {
    matrix.setLed(0, row, col, gameboard[row][col] == 0 ? 0 : 1);}}
    delay(50);}
    delay(600);
 for (int i = 1; i <= snakeLength; i++) {
  for (int row = 0; row < 8; row++) {
   for (int col = 0; col < 8; col++) {
    if (gameboard[row][col] == i) {
     matrix.setLed(0, row, col, 0);
     delay(100);}}}}}
```

```
void calibrateJoystick() {
 Coordinate values;
 for (int i = 0; i < 10; i++) {
  values.x += analogRead(Pin::joystickX);
  values.y += analogRead(Pin::joystickY);}
 joystickHome.x = values.x / 10;
 joystickHome.y = values.y / 10;}
void initialize() {
 pinMode(Pin::joystickVCC, OUTPUT);
 digitalWrite(Pin::joystickVCC, HIGH);
 pinMode(Pin::joystickGND, OUTPUT);
 digitalWrite(Pin::joystickGND, LOW);
 matrix.shutdown(0, false);
 matrix.setIntensity(0, intensity);
 matrix.clearDisplay(0);
 randomSeed(analogRead(A5));
 snake.row = random(8);
 snake.col = random(8);
}
void dumpGameBoard() {
 String buff = "\n\n\n";
 for (int row = 0; row < 8; row++) {
  for (int col = 0; col < 8; col++) {
   if (gameboard[row][col] < 10) buff += " ";
   if (gameboard[row][col] != 0) buff += gameboard[row][col];
   else if (col == food.col && row == food.row) buff += "@";
   else buff += "-";
   buff += " ";
  }
  buff += "\n";
 }
 Serial.println(buff);
```

```
void showSnakeMessage() {
  [&] {
    for (int d = 0; d < sizeof(snakeMessage[0]) - 7; d++) {
      for (int col = 0; col < 8; col++) {
        delay(messageSpeed);
        for (int row = 0; row < 8; row++) {
          // this reads the byte from the PROGMEM and displays it on the screen
          matrix.setLed(0, row, col, pgm_read_byte(&(snakeMessage[row][col + d])));}}
      if (analogRead(Pin::joystickY) < joystickHome.y - joystickThreshold
          || analogRead(Pin::joystickY) > joystickHome.y + joystickThreshold
          || analogRead(Pin::joystickX) < joystickHome.x - joystickThreshold
          || analogRead(Pin::joystickX) > joystickHome.x + joystickThreshold) {
        return; }}
  }();
  matrix.clearDisplay(0);
  while (analogRead(Pin::joystickY) < joystickHome.y - joystickThreshold
         || analogRead(Pin::joystickY) > joystickHome.y + joystickThreshold
         || analogRead(Pin::joystickX) < joystickHome.x - joystickThreshold
         || analogRead(Pin::joystickX) > joystickHome.x + joystickThreshold) {}}
```

```
void showGameOverMessage() {
  [&] {
    for (int d = 0; d < sizeof(gameOverMessage[0]) - 7; d++) {
      for (int col = 0; col < 8; col++) {
        delay(messageSpeed);
        for (int row = 0; row < 8; row++) {
          matrix.setLed(0, row, col, pgm_read_byte(&(gameOverMessage[row][col + d])));}}
      if (analogRead(Pin::joystickY) < joystickHome.y - joystickThreshold
          || analogRead(Pin::joystickY) > joystickHome.y + joystickThreshold
          || analogRead(Pin::joystickX) < joystickHome.x - joystickThreshold
          || analogRead(Pin::joystickX) > joystickHome.x + joystickThreshold) {
        return;}}
  }();
  matrix.clearDisplay(0);
  while (analogRead(Pin::joystickY) < joystickHome.y - joystickThreshold
         || analogRead(Pin::joystickY) > joystickHome.y + joystickThreshold
         || analogRead(Pin::joystickX) < joystickHome.x - joystickThreshold
         || analogRead(Pin::joystickX) > joystickHome.x + joystickThreshold) {}}
void showWinMessage() {
}
```

```
void showScoreMessage(int score) {
  if (score < 0 || score > 99) return;
  int second = score % 10;
  int first = (score / 10) % 10;
  [&] {
    for (int d = 0; d < sizeof(scoreMessage[0]) + 2 * sizeof(digits[0][0]); d++) {
      for (int col = 0; col < 8; col++) {
        delay(messageSpeed);
        for (int row = 0; row < 8; row++) {
          if (d <= sizeof(scoreMessage[0]) - 8) {
            matrix.setLed(0, row, col, pgm_read_byte(&(scoreMessage[row][col + d])));
          }
          int c = col + d - sizeof(scoreMessage[0]) + 6;
          if (score < 10) c += 8;
          if (c >= 0 && c < 8) {
            if (first > 0) matrix.setLed(0, row, col, pgm_read_byte(&(digits[first][row][c]))); // show only if score is >= 10 (see above)
          } else {
            c -= 8;
            if (c >= 0 && c < 8) {
              matrix.setLed(0, row, col, pgm_read_byte(&(digits[second][row][c]))); }}}}
      if (analogRead(Pin::joystickY) < joystickHome.y - joystickThreshold
          || analogRead(Pin::joystickY) > joystickHome.y + joystickThreshold
          || analogRead(Pin::joystickX) < joystickHome.x - joystickThreshold
          || analogRead(Pin::joystickX) > joystickHome.x + joystickThreshold) { return; }}
  }();
  matrix.clearDisplay(0);}
float mapf(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```