Cyclone das Spiel

Die kalte bzw. nasse Jahreszeit beginnt wieder und einmal mehr sucht man Beschäftigung für sein(e) Kind(er) oder will sein Nerd-Können den Kumpels beweisen. Im Internet bin ich dabei auf das Spiel Cyclone, was zu Deutsch Tornado heißt, gestoßen. Dieses Spiel hat mich und meine Kinder fasziniert. Die gezeigten Varianten haben mir aber alle nicht gefallen, kurzerhand wurde eine eigene Variante entwickelt.

Worum geht es in dem Spiel Cyclone

Wenn Sie sich das Video (https://www.youtube.com/watch?v=5HFTi8qphDA&feature=emb_logo) ansehen, ist das Prinzip schnell erklärt. Sie haben einen LED-Ring mit einem laufenden LED-Punkt und einer LED-Zielmarkierung. Der Spielerpunkt läuft mit einer definierten Geschwindigkeit über den Ring und Sie müssen versuchen, sobald der Spielerpunkt mit der Zielmarkierung deckungsgleich ist, eine Taste zu drücken. Anders als in dem Video, habe ich keine Level, sondern ein fortlaufendes Spiel entwickelt, das endet, wenn einmal nicht die Zielmarkierung getroffen wird. Zusätzlich ist ein LCD-Display verbaut, welches den Highscore, den aktuellen Score und die aktuelle Runde anzeigt. Die Geschwindigkeit wird bei jeder neuen Runde zufällig gewählt.

Die Hardware

Bei der benötigten Hardware brauchen Sie nur wenige Teile. Da der Code aber so geschrieben ist, dass Sie den WS2812B-LED-Ring von Arduino auch durch ein WS2812B-Strip austauschen können, werden hier zwei Stücklisten geführt.
Für die Variante, die in diesem Blogbeitrag vorgestellt wird, brauche Sie die Bauteile die in Tabelle 1 aufgelistet sind.

Pos	Anzahl	Bauteil	Link
1	1	Arduino Nano	https://az-delivery.de
2	1	LED Ring 5V RGB WS2812B 12-Bit 50mm	https://az-delivery.de
3	1	Taster	https://az-delivery.de
4	1	LCD-Modul mit i2c-Schnittstelle	https://az-delivery.de
5	1	Steckbrett und Jumper (Hier im Set mit Netzteiladapter)	https://az-delivery.de

Tabelle 1: Hardwareteile für Cyclone mit WS2812B-LEDring

Wollen Sie hingegen einen Strip benutzen, brauchen Sie die Bauteile aus Tabelle 2. Die weiteren Bauteile, um den Strip in einen Rahmen oder Ähnliches zu verbauen, werden dabei nicht berücksichtigt.

Pos	Anzahl	Bauteil	Link
1	1	Arduino Nano	https://az-delivery.de
2	1	WS2812B LED-Strip	https://amazon.de/
3	1	Taster	https://az-delivery.de
4	1	LCD-Modul mit i2c-Schnittstelle	https://az-delivery.de
5	1	Steckbrett und Jumper	https://az-delivery.de

		(Hier im Set mit Netzteiladapter)	
6	1	Netzteil für LEDs und Arduino	https://amazon.de

Tabelle 2: Hardwareteile für Cyclone mit WS2812B-Strip

Gleich hier sollte erwähnt werden, je mehr LEDs ihr WS2812B-Strip hat, umso mehr Strom muss das Netzteil liefern. Für 60 LEDs brauchen Sie, wenn alle LEDs leuchten knapp 4A.

Benötigte Software

Die benötigte Software für dieses Projekt ist überschaubar:

- Arduino IDE (https://www.arduino.cc/en/Main/Software), hier laden Sie am besten die aktuelle Version herunter
- Die Bibliothek **Adafruit NeoPixel**, bei Erstellung vom Beitrag in Version 1.6.0 verfügbar, mit allen Abhängigkeiten, die Sie über die Bibliotheksverwaltung, siehe https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/arduino-ide-programmieren-fuer-einsteiger-teil-1 Abschnitt Bibliotheksverwaltung, installieren müssen.

Der Aufbau

Für den Aufbau mit dem WS2812B-Ring müssen die Bauteile, wie in Abbildung 1 gezeigt, miteinander verbunden werden. Haben Sie einen WS2812B-Strip, so müssen Sie die Spannungsversorgung und den Data-IN-Anschluss korrekt anschließen.

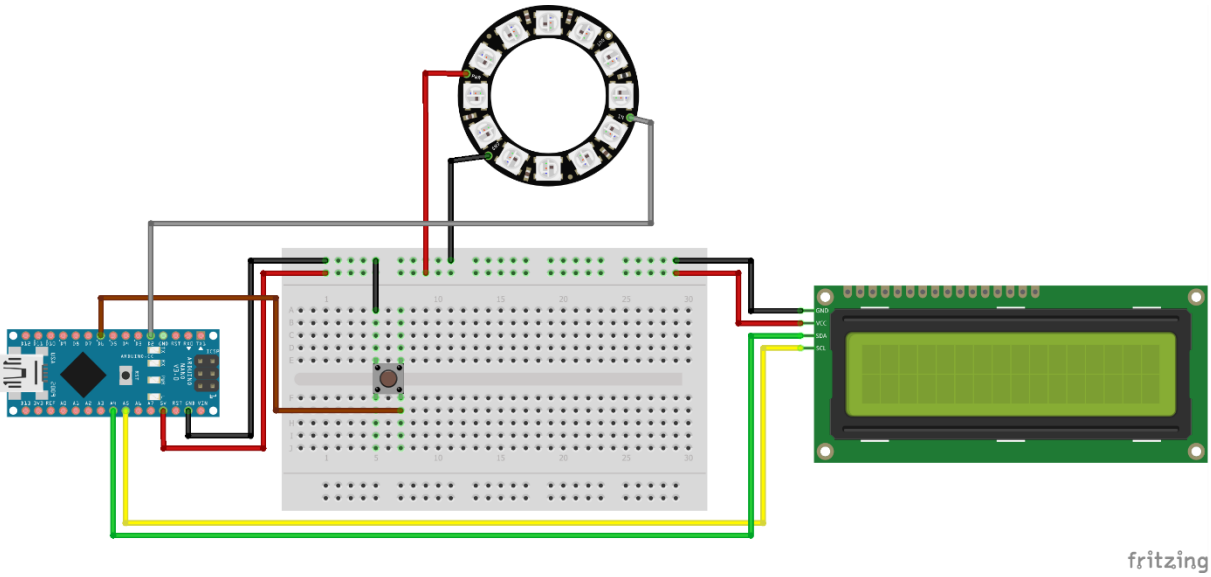

Abbildung 1: Verdrahtung der einzelnen Komponenten

Für beide WS2812B-Varianten gilt, rot ist die Phase (5 Volt), schwarz die Masse (GND) und grau der Data-IN-Anschluss. In den meisten Fällen haben Sie vier Anschlüsse bei WS2812B-Strips, daher sollten Sie im Vorfeld prüfen, welcher der richtige Data-Pin ist.

Der Quellcode

Entweder kopieren sie den Code hier aus dem Blog, siehe //----------
// Game "CYCLONE" for Arduino
// Autor: Joern Weise

```
// License: GNU GPl 3.0
// Created: 20. Sep 2020
// Update:  25. Sep 2020
//------------------------------------------------------
//Include libraries
#include <Adafruit_NeoPixel.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>

//Defines
#define NUMPIXELS   12  // Popular NeoPixel ring size or edit the number of LEDs
#define PIN         2   // Data-Pin to ring or strip
#define PINBTN      6   // Pin for Player-button
#define PINSCORERST 9   // Pin to reset score during first run

#define DISABLEWINDOW 3 //Rounds before the LED before and after target is not valid anymore

//Player-Dot speed defines
#define STARTINTERVAL 250 //"Normal" move
#define MAXINTERVAL 500   //Very slow move
#define MININTERVAL 50    //Very fast move

//Create objects
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD adress
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800); //Init NeoPixel object

bool bFirstRun, bSecureWindow;
int iState = 1;
int iTargetPos, iPlayerPos, iStoredHighscore, iRound, iScore, iInterval;  //Vars for the game
int iLastButtonPressed, iButtonState, iDebounceButton;  //Vars to debounce button
unsigned long iLastPlayerMove, ulLastDebounceTime;      //Timer to debouce button
unsigned long ulDebounceButton = 10;                    //Debounce-time

void setup() {
  Serial.begin(115200);
  Serial.println("Init serial communication: DONE");

  //Begin init for WS218B-ring or -strip
  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
  pixels.clear(); // Set all pixels to "off"
  pixels.setBrightness(20); // Set brightness to 20%
  pixels.show();   // Send the updated pixel colors to the hardware.
  Serial.println("Init WS218B-ring: DONE");

  //Begin init display
  lcd.init();
  lcd.backlight();
  lcd.clear();
  Serial.println("Init LCD display: DONE");

  randomSeed(analogRead(0));  // Make randome more randome
  Serial.println("Make randome more randome: DONE");
```

```
  //Read latest highscore from EEPROM
  iStoredHighscore = EEPROM.read(0);
  Serial.println("Last stored highscore: " + String(iStoredHighscore));

  //Init button with internal pullup-resistor
  pinMode(PINBTN,INPUT_PULLUP); //GameBTN
  pinMode(PINSCORERST,INPUT_PULLUP); //Reset-Pin for score

  //Init some basic-vars
  bFirstRun = true; //Enable firstrun
  iLastButtonPressed = digitalRead(PINBTN); //Init iLastButtonPressed
  iButtonState = digitalRead(PINBTN); //Init iButtonstate
}

void loop() {
  int iDebounceButton = DebounceButton(); //Check and debounce button

  if(!bFirstRun)
  {
    if(iState == 1) //Startscreen
    {
      bSecureWindow = true;
      iRound = 1;
      iScore = 0;
      iInterval = STARTINTERVAL;
      lcd.clear();
      lcd.home();
      lcd.print("Highscore: " + String(iStoredHighscore));
      lcd.setCursor(0,1);
      lcd.print("Press button ...");
      iState = 2;
    }
    if(iState == 2) //Get Button pressed
    {
      if(iDebounceButton == LOW)
      {
        if(iRound == 1) //Only show once during game
          Serial.println("-------- New game --------");
        lcd.clear();
        lcd.home();
        lcd.print("Release button");
        lcd.setCursor(0,1);
        lcd.print("to start");
        iState = 3;
      }
    }
    if(iState == 3) //Init next round
    {
      if(iDebounceButton == HIGH)
      {
        lcd.clear();
        lcd.home();
```

```
    lcd.print("Round: " + String(iRound));
    Serial.println("Round: " + String(iRound));
    lcd.setCursor(0,1);
    lcd.print("Score: " + String(iScore));
    Serial.println("Score: " + String(iScore));
    iTargetPos = random(0,NUMPIXELS-1);
    Serial.println("New target pos: " + String(iTargetPos));
    iPlayerPos = random(0,NUMPIXELS-1);
    while(iTargetPos == iPlayerPos)
      iPlayerPos = random(0,NUMPIXELS-1);
    Serial.println("Player start pos: " + String(iPlayerPos));
    iState = 4;
  }
 }
 if(iState == 4) //Draw target and playes dot
 {
  DrawNextTarget(iTargetPos, bSecureWindow);  //Draw new target
  DrawPlayer(iPlayerPos);                 //Draw player dot
  iLastPlayerMove = millis();             //Update timer for moving
  iState = 5;
 }
 if(iState == 5) //Wait pressing button and move player dot
 {
  if(iDebounceButton == LOW)
  {
    iState = 6;
  }
  else
  {
    unsigned long currentMillis = millis();
    if(currentMillis - iLastPlayerMove > iInterval)
    {
      iPlayerPos++;
      if(iPlayerPos >= NUMPIXELS)
        iPlayerPos = 0;
      DrawNextTarget(iTargetPos, bSecureWindow);
      DrawPlayer(iPlayerPos);
      iLastPlayerMove = currentMillis;
    }
  }
 }
 if(iState == 6) //Check if player win
 {
  if(CheckPlayerPos())  //Winner or loser?
  {
    iScore++; //Update score
    iRound++; //Update rounds
    iState = 2; //Go back to release button
    if(iRound > DISABLEWINDOW)  //Only target
    {
      bSecureWindow = false;
      iInterval = random(MININTERVAL,MAXINTERVAL);
    }
```

```
        else
          iInterval = random(STARTINTERVAL-50,MAXINTERVAL);
        Serial.println("New interval: " + String(iInterval));
       }
       else
        iState = 90;
     }
     if(iState == 90)  //Game ends
     {
      Serial.println("Game ends");
      lcd.clear();
      lcd.home();
      iDebounceButton = HIGH;
      iLastButtonPressed = HIGH;
      iButtonState = HIGH;
      if(iScore > iStoredHighscore) //New highscore?
      {
        lcd.print("New highscore ");
        lcd.setCursor(0,1);
        lcd.print("New score: " + String(iScore));
        Serial.println("New highscore is " + String(iScore));
        EEPROM.write(0,iScore); //Store new highscore to EEPROM
        iStoredHighscore = iScore;
      }
      else  //Loser
      {
        lcd.print("Game Over");
        lcd.setCursor(0,1);
        lcd.print("You lose");
        Serial.println("You lose!");
      }
      Serial.println("-------- End game --------");
      delay(2000);
      iState = 1;
     }
   }
   else
     InitFirstRun(); //Init Firstrun to check LCD and WS218B-ring
}

//Function to make first run
void InitFirstRun()
{
  if(digitalRead(PINSCORERST) == LOW) //Overwrite EEPROM with "0"
  {
    Serial.println("Reset highscore");
    for(int iCnt = 0; iCnt < EEPROM.length(); iCnt++)
      EEPROM.write(iCnt,0);
  }
  Serial.println("---- Start init ----");
  lcd.home();
  lcd.print("Game Cyclone");
  Serial.println("Game Cyclone");
```

```
  lcd.setCursor(0,1);
  lcd.print("(c)  M3taKn1ght");
  Serial.print("(c)  M3taKn1ght");
  delay(1000);
  lcd.clear();
  lcd.home();
  lcd.print("For az-Delivery");
  Serial.println("For az-Delivery");
  lcd.setCursor(0,1);
  lcd.print("Testing ring ...");
  Serial.println("Testing ring ...");
  delay(1000);
  pixels.clear();
  //Check every single LED
  for(int i = 0; i<=255; i+=51)
  {
    InitRingTest(i,0,0);
    delay(50);
  }
  pixels.clear();
  for(int i = 0; i<=255; i+=51)
  {
    InitRingTest(0,i,0);
    delay(50);
  }
  pixels.clear();
  for(int i = 0; i<=255; i+=51)
  {
    InitRingTest(0,0,i);
    delay(50);
  }
  pixels.clear();
  pixels.show();
  Serial.println("----  End init  ----");
  bFirstRun = false;
  Serial.println("bFirstRun: " + String(bFirstRun));
  Serial.println("Activate state for game");
}

//Simple function to check LED-Ring one by one
void InitRingTest(int iRed, int iGreen, int iBlue)
{
  Serial.println("R: " + String(iRed) + " G: " + String(iGreen) + " B: " + String(iBlue));
  for(int iPixel = 0; iPixel < NUMPIXELS; iPixel++)
  {
    pixels.setPixelColor(iPixel, pixels.Color(iRed, iGreen, iBlue));
    pixels.show();
    delay(50);
  }
}

//Function to draw target an secure area for player
void DrawNextTarget(int iPos, bool bArea)
```

```
{
  pixels.clear();
  pixels.setPixelColor(iPos, pixels.Color(0, 255, 0));
  if(bArea)
  {
    if(iPos - 1 < 0)
      pixels.setPixelColor(NUMPIXELS - 1, pixels.Color(255, 136, 0));
    else
      pixels.setPixelColor(iPos - 1, pixels.Color(255, 136, 0));

    if(iPos + 1 >= NUMPIXELS)
      pixels.setPixelColor(0, pixels.Color(255, 136, 0));
    else
      pixels.setPixelColor(iPos + 1, pixels.Color(255, 136, 0));
  }
}

//Function to draw players LED
void DrawPlayer(int iPos)
{
  if(iPos == iTargetPos)  //target and player-dot is equal
    pixels.setPixelColor(iPos, pixels.Color(0, 0, 255));  //Dot color will blue
  else
    pixels.setPixelColor(iPos, pixels.Color(255, 0, 0));  //Otherwise red
  pixels.show();
}

//Function to check after pressing button, if user hit the target
bool CheckPlayerPos()
{
  if(iTargetPos == iPlayerPos)  //Player hit target?
    return true;
  else
  {
    if(bSecureWindow) //LED before and after target active?
    {
      int iBeforeTarget = iTargetPos - 1;
      int iAfterTarget = iTargetPos + 1;
      if(iBeforeTarget < 0)
        iBeforeTarget = NUMPIXELS - 1;
      if(iAfterTarget >= NUMPIXELS)
        iAfterTarget = 0;
      if(iBeforeTarget == iPlayerPos || iAfterTarget == iPlayerPos)
        return true;
      else
        return false;
    }
    else
      return false;
  }
}

//Function to debounce button
```

```
int DebounceButton()
{
  int iCurrentButtonState = digitalRead(PINBTN);
  if(iCurrentButtonState != iLastButtonPressed)
    ulLastDebounceTime = millis();

  if((millis() - ulLastDebounceTime) > ulDebounceButton)
  {
    if(iCurrentButtonState != iButtonState)
      iButtonState = iCurrentButtonState;
  }
  iLastButtonPressed = iCurrentButtonState;
  return iButtonState;
```

Code 1, oder Sie können Ihn über das [Github-Repository vom Autor] runterladen.

```
//------------------------------------------------------
// Game "CYCLONE" for Arduino
// Autor:   Joern Weise
// License: GNU GPl 3.0
// Created: 20. Sep 2020
// Update:  25. Sep 2020
//------------------------------------------------------
//Include libraries
#include <Adafruit_NeoPixel.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>

//Defines
#define NUMPIXELS   12  // Popular NeoPixel ring size or edit the number of LEDs
#define PIN         2   // Data-Pin to ring or strip
#define PINBTN      6   // Pin for Player-button
#define PINSCORERST 9   // Pin to reset score during first run

#define DISABLEWINDOW 3 //Rounds before the LED before and after target is not valid anymore

//Player-Dot speed defines
#define STARTINTERVAL 250 //"Normal" move
#define MAXINTERVAL 500   //Very slow move
#define MININTERVAL 50    //Very fast move

//Create objects
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD adress
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800); //Init NeoPixel object

bool bFirstRun, bSecureWindow;
int iState = 1;
int iTargetPos, iPlayerPos, iStoredHighscore, iRound, iScore, iInterval;  //Vars for the game
int iLastButtonPressed, iButtonState, iDebounceButton;  //Vars to debounce button
unsigned long iLastPlayerMove, ulLastDebounceTime;      //Timer to debouce button
unsigned long ulDebounceButton = 10;                    //Debounce-time

void setup() {
```

```
  Serial.begin(115200);
  Serial.println("Init serial communication: DONE");

  //Begin init for WS218B-ring or -strip
  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
  pixels.clear(); // Set all pixels to "off"
  pixels.setBrightness(20); // Set brightness to 20%
  pixels.show();   // Send the updated pixel colors to the hardware.
  Serial.println("Init WS218B-ring: DONE");

  //Begin init display
  lcd.init();
  lcd.backlight();
  lcd.clear();
  Serial.println("Init LCD display: DONE");

  randomSeed(analogRead(0));  // Make randome more randome
  Serial.println("Make randome more randome: DONE");

  //Read latest highscore from EEPROM
  iStoredHighscore = EEPROM.read(0);
  Serial.println("Last stored highscore: " + String(iStoredHighscore));

  //Init button with internal pullup-resistor
  pinMode(PINBTN,INPUT_PULLUP); //GameBTN
  pinMode(PINSCORERST,INPUT_PULLUP); //Reset-Pin for score

  //Init some basic-vars
  bFirstRun = true; //Enable firstrun
  iLastButtonPressed = digitalRead(PINBTN); //Init iLastButtonPressed
  iButtonState = digitalRead(PINBTN); //Init iButtonstate
}

void loop() {
  int iDebounceButton = DebounceButton(); //Check and debounce button

  if(!bFirstRun)
  {
    if(iState == 1) //Startscreen
    {
      bSecureWindow = true;
      iRound = 1;
      iScore = 0;
      iInterval = STARTINTERVAL;
      lcd.clear();
      lcd.home();
      lcd.print("Highscore: " + String(iStoredHighscore));
      lcd.setCursor(0,1);
      lcd.print("Press button ...");
      iState = 2;
    }
    if(iState == 2) //Get Button pressed
    {
```

```
  if(iDebounceButton == LOW)
  {
    if(iRound == 1) //Only show once during game
      Serial.println("-------- New game --------");
    lcd.clear();
    lcd.home();
    lcd.print("Release button");
    lcd.setCursor(0,1);
    lcd.print("to start");
    iState = 3;
  }
}
if(iState == 3) //Init next round
{
  if(iDebounceButton == HIGH)
  {
    lcd.clear();
    lcd.home();
    lcd.print("Round: " + String(iRound));
    Serial.println("Round: " + String(iRound));
    lcd.setCursor(0,1);
    lcd.print("Score: " + String(iScore));
    Serial.println("Score: " + String(iScore));
    iTargetPos = random(0,NUMPIXELS-1);
    Serial.println("New target pos: " + String(iTargetPos));
    iPlayerPos = random(0,NUMPIXELS-1);
    while(iTargetPos == iPlayerPos)
      iPlayerPos = random(0,NUMPIXELS-1);
    Serial.println("Player start pos: " + String(iPlayerPos));
    iState = 4;
  }
}
if(iState == 4) //Draw target and playes dot
{
  DrawNextTarget(iTargetPos, bSecureWindow);  //Draw new target
  DrawPlayer(iPlayerPos);                     //Draw player dot
  iLastPlayerMove = millis();                 //Update timer for moving
  iState = 5;
}
if(iState == 5) //Wait pressing button and move player dot
{
  if(iDebounceButton == LOW)
  {
    iState = 6;
  }
  else
  {
    unsigned long currentMillis = millis();
    if(currentMillis - iLastPlayerMove > iInterval)
    {
      iPlayerPos++;
      if(iPlayerPos >= NUMPIXELS)
        iPlayerPos = 0;
```

```
      DrawNextTarget(iTargetPos, bSecureWindow);
      DrawPlayer(iPlayerPos);
      iLastPlayerMove = currentMillis;
    }
   }
  }
  if(iState == 6) //Check if player win
  {
   if(CheckPlayerPos())  //Winner or loser?
   {
    iScore++; //Update score
    iRound++; //Update rounds
    iState = 2; //Go back to release button
    if(iRound > DISABLEWINDOW)  //Only target
    {
     bSecureWindow = false;
     iInterval = random(MININTERVAL,MAXINTERVAL);
    }
    else
     iInterval = random(STARTINTERVAL-50,MAXINTERVAL);
    Serial.println("New interval: " + String(iInterval));
   }
   else
    iState = 90;
  }
  if(iState == 90)  //Game ends
  {
   Serial.println("Game ends");
   lcd.clear();
   lcd.home();
   iDebounceButton = HIGH;
   iLastButtonPressed = HIGH;
   iButtonState = HIGH;
   if(iScore > iStoredHighscore) //New highscore?
   {
    lcd.print("New highscore ");
    lcd.setCursor(0,1);
    lcd.print("New score: " + String(iScore));
    Serial.println("New highscore is " + String(iScore));
    EEPROM.write(0,iScore); //Store new highscore to EEPROM
    iStoredHighscore = iScore;
   }
   else  //Loser
   {
    lcd.print("Game Over");
    lcd.setCursor(0,1);
    lcd.print("You lose");
    Serial.println("You lose!");
   }
   Serial.println("-------- End game --------");
   delay(2000);
   iState = 1;
  }
```

```
  }
  else
    InitFirstRun(); //Init Firstrun to check LCD and WS218B-ring
}

//Function to make first run
void InitFirstRun()
{
  if(digitalRead(PINSCORERST) == LOW) //Overwrite EEPROM with "0"
  {
    Serial.println("Reset highscore");
    for(int iCnt = 0; iCnt < EEPROM.length(); iCnt++)
      EEPROM.write(iCnt,0);
  }
  Serial.println("---- Start init ----");
  lcd.home();
  lcd.print("Game Cyclone");
  Serial.println("Game Cyclone");
  lcd.setCursor(0,1);
  lcd.print("(c)  M3taKn1ght");
  Serial.print("(c)  M3taKn1ght");
  delay(1000);
  lcd.clear();
  lcd.home();
  lcd.print("For az-Delivery");
  Serial.println("For az-Delivery");
  lcd.setCursor(0,1);
  lcd.print("Testing ring ...");
  Serial.println("Testing ring ...");
  delay(1000);
  pixels.clear();
  //Check every single LED
  for(int i = 0; i<=255; i+=51)
  {
    InitRingTest(i,0,0);
    delay(50);
  }
  pixels.clear();
  for(int i = 0; i<=255; i+=51)
  {
    InitRingTest(0,i,0);
    delay(50);
  }
  pixels.clear();
  for(int i = 0; i<=255; i+=51)
  {
    InitRingTest(0,0,i);
    delay(50);
  }
  pixels.clear();
  pixels.show();
  Serial.println("----  End init  ----");
  bFirstRun = false;
```

```
  Serial.println("bFirstRun: " + String(bFirstRun));
  Serial.println("Activate state for game");
}

//Simple function to check LED-Ring one by one
void InitRingTest(int iRed, int iGreen, int iBlue)
{
  Serial.println("R: " + String(iRed) + " G: " + String(iGreen) + " B: " + String(iBlue));
  for(int iPixel = 0; iPixel < NUMPIXELS; iPixel++)
  {
    pixels.setPixelColor(iPixel, pixels.Color(iRed, iGreen, iBlue));
    pixels.show();
    delay(50);
  }
}

//Function to draw target an secure area for player
void DrawNextTarget(int iPos, bool bArea)
{
  pixels.clear();
  pixels.setPixelColor(iPos, pixels.Color(0, 255, 0));
  if(bArea)
  {
    if(iPos - 1 < 0)
      pixels.setPixelColor(NUMPIXELS - 1, pixels.Color(255, 136, 0));
    else
      pixels.setPixelColor(iPos - 1, pixels.Color(255, 136, 0));

    if(iPos + 1 >= NUMPIXELS)
      pixels.setPixelColor(0, pixels.Color(255, 136, 0));
    else
      pixels.setPixelColor(iPos + 1, pixels.Color(255, 136, 0));
  }
}

//Function to draw players LED
void DrawPlayer(int iPos)
{
  if(iPos == iTargetPos)  //target and player-dot is equal
    pixels.setPixelColor(iPos, pixels.Color(0, 0, 255));  //Dot color will blue
  else
    pixels.setPixelColor(iPos, pixels.Color(255, 0, 0));  //Otherwise red
  pixels.show();
}

//Function to check after pressing button, if user hit the target
bool CheckPlayerPos()
{
  if(iTargetPos == iPlayerPos)  //Player hit target?
    return true;
  else
  {
    if(bSecureWindow) //LED before and after target active?
```

```
   {
    int iBeforeTarget = iTargetPos - 1;
    int iAfterTarget = iTargetPos + 1;
    if(iBeforeTarget < 0)
     iBeforeTarget = NUMPIXELS - 1;
    if(iAfterTarget >= NUMPIXELS)
     iAfterTarget = 0;
    if(iBeforeTarget == iPlayerPos || iAfterTarget == iPlayerPos)
     return true;
    else
     return false;
   }
   else
    return false;
  }
}

//Function to debounce button
int DebounceButton()
{
 int iCurrentButtonState = digitalRead(PINBTN);
 if(iCurrentButtonState != iLastButtonPressed)
  ulLastDebounceTime = millis();

 if((millis() - ulLastDebounceTime) > ulDebounceButton)
 {
  if(iCurrentButtonState != iButtonState)
   iButtonState = iCurrentButtonState;
 }
 iLastButtonPressed = iCurrentButtonState;
 return iButtonState;
```

Code 1: Code zum Spiel "Cyclone"

Sie können an dieser Stelle einfach den Code via Arduino IDE hochladen, jedoch sollen noch einige Stellen vom Code näher erläutert werden.

Damit Sie das Display und die WS2812B-LED ansteuern können, müssen zunächst für beide ein entsprechendes Objekt angelegt werden. Dies sehen Sie direkt am Anfang von Code 1, hinter dem Kommentar „Create object“. Direkt danach werden noch einige globale Variablen erzeugt und teilweise, sofern das nicht direkt danach in der setup()-Funktion geschieht, initialisiert. Für Sie vielleicht interessant ist die Zeile „iStoredHighscore = EEPROM.read(0);“ bei welcher der letzte gespeicherte Wert vom Highscore aus dem EEPROM, also dem Speicher der seine Werte bei Spannungslosigkeit nicht verliert, gelesen wird. Sofern der Highscore während eines Spiels überboten wurde, wird in der loop()-Funktion mittels der Zeile „EEPROM.write(0,iScore);“, der neue Highscore in den EEPROM geschrieben.

Die InitFirstRun()-Funktion wird in dem Code nur dann aufgerufen, wenn der Arduino neu gestartet wird. Wollen Sie einen Highscore oder alte Werte aus dem EEPROM löschen, so geschieht das in dieser Funktion. Verbinden Sie dazu noch vor dem Start des Arduino Nanos den Digitalpin 9 mit GND. Durch dieses Vorgehen ist der EEPROM komplett auf null gesetzt, bevor alle Farben von allen LEDs geprüft werden. Für diesen Test der LEDs ist es wichtig, dass Sie eine geeignete Spannungsversorgung für Ihre Schaltung haben.

Die loop()-Funktion ist das Herzstück des Spiels. Zum einen wird direkt am Anfang der aktuelle Zustand vom Drucktaster ermittelt und bei einer Betätigung dieser entprellt. Entprellt bedeutet, dass für eine definierte Zeit, in diesem Fall für 10 ms, ein eindeutiger Signalwechsel vorhanden sein muss. Mehr zum Thema entprellen könne Sie hier nachlesen. Direkt danach wird der Ablauf des Spiels gesteuert. Welche Ausgaben müssen auf dem Screen visualisiert und ggf. welche LEDs auf dem WS2812B-Ring angezeigt werden.

Die Visualisierung des Zielpunkts und des Spielerpunkts ist über die Funktionen DrawNextTarget(int iPos, bool bArea) bzw. DrawPlayer(int iPos) umgesetzt worden. Diese Funktionen werden, sobald die Zeit für den nächsten Schritt vom Spielerpunkt erreicht ist, mit der LED-Position aufgerufen. Damit auch in den ersten Runden noch die LEDs vor und nach dem Zielpunkt angezeigt werden, wird ein bool-Flag der Funktion DrawNextTarget(int iPos, bool bArea) übergeben. Betätigt nun der Spieler den Drucktaster und das Entprellen ist abgeschlossen, so wird die Funktion bool CheckPlayerPos() aufgerufen. Diese Funktion prüft, ob der Spieler im richtigen Moment den Drucktaster gedrückt hat oder nicht. In den ersten Runden, wenn auch die LED vor und hinter dem Zielpunkt noch gültig ist, wird die Toleranzzone noch berücksichtigt. Hat der Spieler das Ziel erwischt, so wird der aktuelle Score erhöht, eine neue, zufällige Geschwindigkeit ermittelt und die Position vom Zielpunkt und die Startposition vom Spielerpunkt gesetzt. Dabei kann es nicht vorkommen, dass Startpunkt und Zielpunkt direkt am Anfang vom Spiel identisch sind.
Hat der Spieler allerdings zur falschen Zeit gedrückt, wird geprüft, ob der Highscore überboten wurde und ein „Game over“-Screen angezeigt. Bei überbotenen Highscore wird der neue Score direkt in den EEPROM geschrieben.

Damit Sie schneller den Code verstehen und ggf. auch Modifikationen für ein WS2818B-Strip umsetzen können, sind viele Kommentare eingefügt worden. Natürlich können Sie auch weitere Effekte für den WS2812B-Ring vor dem Spiel oder beim „Game Over“ einprogrammieren. Der Code soll erst einmal die Basis für Ihre individuellen Anpassungen sein.

Ich wünsche Ihnen viel Spaß beim Nachbau.
Dieses und weitere Projekte finden sich auf GitHub unter https://github.com/M3taKn1ght/Blog-Repo.