

## Unsere Discobrinne gibt Feedback!

Hallo und herzlich willkommen zum letzten Teil unserer Disco Brillen Reihe.

Im heutigen Teil werden wir nochmals alle relevanten Informationen für den Aufbau der Brille zusammenfassen, ein paar Bilder der Animationssequenzen zeigen und natürlich unserer Brille wieder mal ein Funktionsupgrade spendieren. Das Funktionsupgrade besteht im heutigen Teil aus einem Vibrationssensor, der durch deine tanzenden Bewegungen in der Disco aktiviert wird und damit unserem Micro Controller Informationen über dein „Tanzverhalten“ auf der Tanzfläche gibt. Aus diesen Informationen werden wiederum die Leuchtsequenzen der Brille gesteuert.

Dies bedeutet anders gesagt, deine Brille reagiert auf deine rhythmischen Tanzbewegungen! Nun, wie funktioniert das ganze technisch?

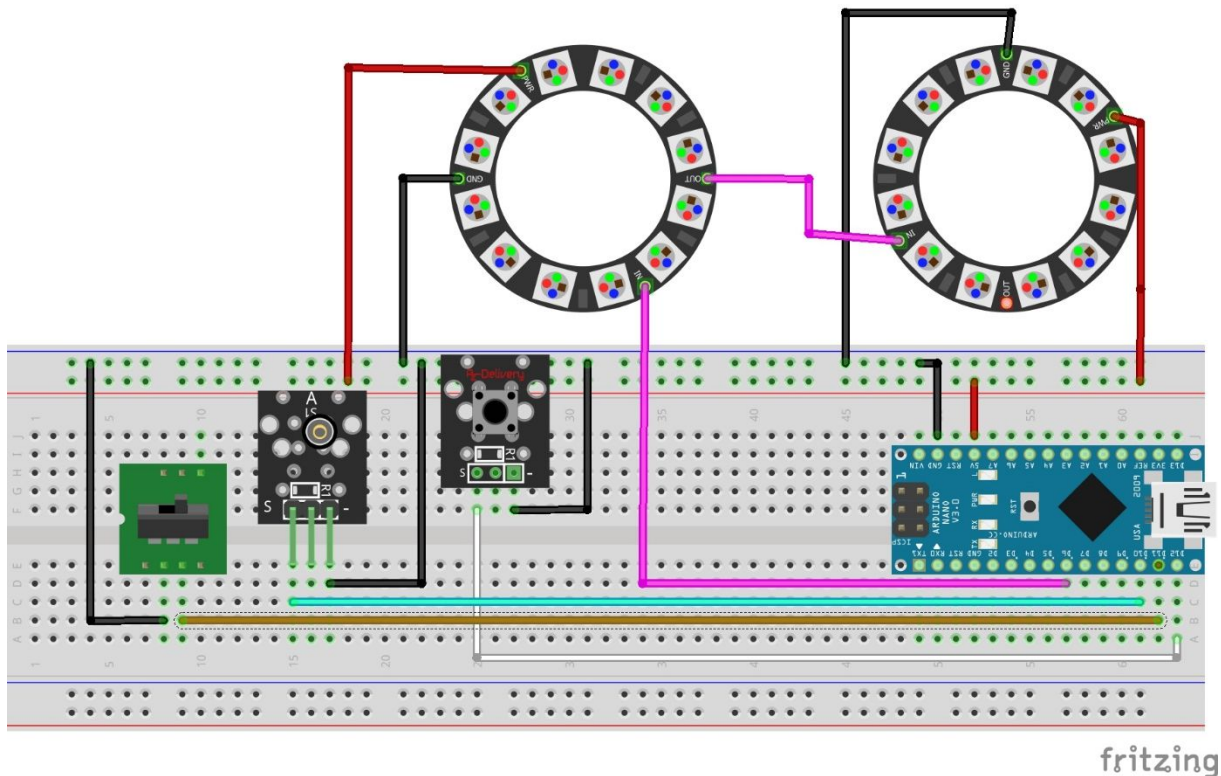
Als Kernelement benötigen wir zunächst einmal einen Vibrationssensor. Diesen gibt es bei uns im Shop in gleich mehreren Ausführungen. Jeder der genannten Sensoren reagiert etwas anders auf Bewegungen. Somit wird das Endergebnis natürlich auch von dem verwendeten Sensor abhängen. Ich empfehle das [KY-002n Sensor Modul](#) als Vibrationssensor zu verwenden. Dieses habe ich auch bei meinen eigenen Versuchsaufbauten und beim Entwickeln des Codes eingesetzt. Wichtig ist, dass der Sensor aktiv nach Masse schaltet. Dies bedeutet, dass bei einem „Schock“ der verwendete Sensor den Kontakt nach GND kurzzeitig schließt.

**Tipp:** Probiere gegebenenfalls verschiedene eigene Sensoren aus, um dein bestes Ergebnis zu bekommen. Eine Auswahl an alternativen Sensoren findest du am Ende dieses Blogs!

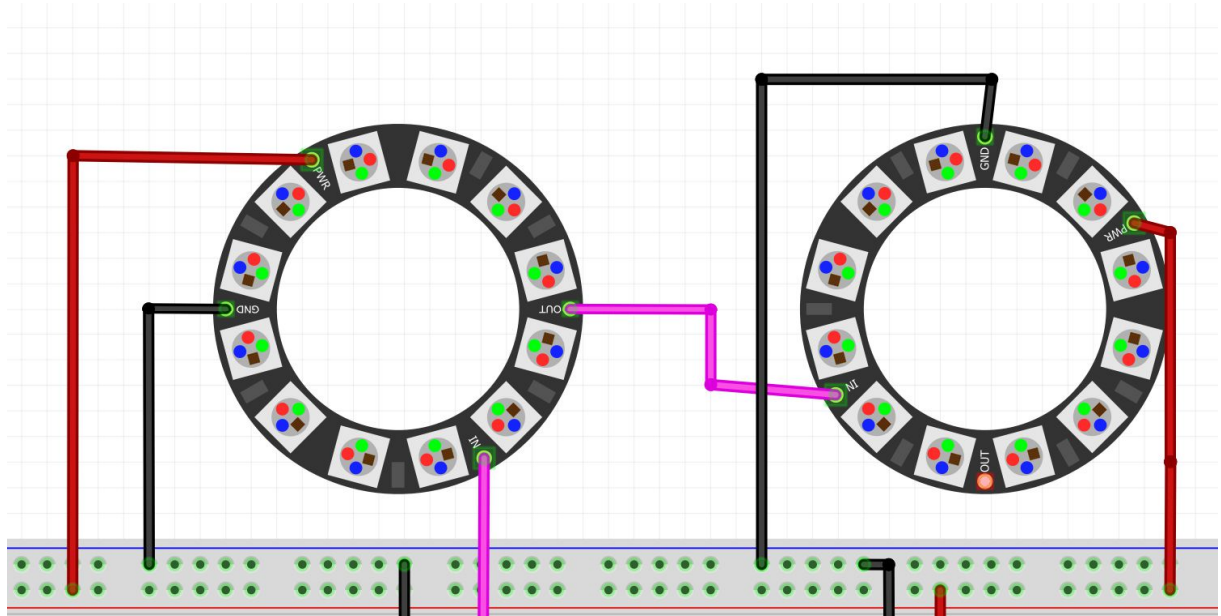
Als zweite neue Hardwarekomponente wird noch ein einfacherer Schalter benötigt. Dieser Schalter dient zum Umschalten zwischen dem alten autarken, aus Teil 2 bekannten, Modus auf den neuen, auf den Bewegungen reagierenden Modus. Die Umschaltung zwischen den Modis kann dabei zu einem beliebigen Zeitpunkt erfolgen. Auch beispielsweise während eines laufenden Programmes!

Kommen wir aber nun zur Verdrahtung der beiden neuen Bauteile in unserer Brille.

Wie auf dem Schaltbild zu erkennen ist, sitzt der Vibrationssensor zwischen GND und Port 10 und der Schalter zwischen GND und Port 11 des Prozessors. Die Verdrahtung des Tasters hat sich dagegen nicht geändert:



Auch die Verdrahtung der beiden WS2812 Ringe hat sich nicht gegenüber dem vorherigen Teil der Reihe verändert:



Bitte ladet nach der Hardwareaktualisierung den folgenden Funktionsaktualisierten Code auf eure Brille hoch:

```
#include <Adafruit_NeoPixel.h>

#define BUTTON_CHANGEANIMATION 12 // Digital IO pin connected to the
button. This will be
// driven with a pull-up resistor so the switch should
// pull the pin to ground momentarily. On a high -> low
// transition the button press logic will execute.
#define PIXEL_PIN 6 // Digital IO pin connected to the NeoPixels.
#define SHOCK_SENSOR 10 // Shock / Vibration Sensor attached
#define MODE_SWITCH 11 // Operation Mode
#define PIXEL_COUNT 24 // All Pixels on Strip
#define MaxAninmationsAvail 4

Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN,
NEO_RGB + NEO_KHZ800);

const int hueRedLow = 0;
const int hueRedHigh = 255;
const int hueBlue = 170;
const int angleMin = 0;
const int angleSector = 60;
const int angleMax = 360;
const int brightMin = 0;
const int brightMax = 255;

byte hue, brightness;
// The saturation is fixed at 255 (full) to remove bleed-through of different
// colours.
byte saturation = 255;
// interrut Control
bool A60telSecInterruptOccured = true;
bool FunctionMode = false;
byte A60telSeconds24 = 0;
byte A4telSeconds24 = 0;
// Timer Variables
int TimerSeconds = 0; // Counter
int TimerAlarmSet = 15; // 15 Second Timer
bool TimerStartFlagFlag = false;
bool TimerStop = true;
//Manual Operations
bool ButtonAPress = false;

//AnimationControl
int ShouldAnimation = 0;
int IsAnimation = 0;
int OLDLightBorder = 0;
bool GetONOFFStatus = false;
```

```

bool OLDONOFFStatus = false;
bool PlayIntro = false; //Play Intro
bool PlayOutro = false; //Play Outro
bool ChangeAnimation = false;
bool RunOnce = true; // Power Off Animation - Animation 0

//universal variables
byte a,c,d,e,f;
unsigned int r, g, b;

//Interrupt Routines

ISR(TIMER1_COMPA_vect)
{
    bool LEDChange,PressedZ;
    PressedZ= digitalRead(BUTTON_CHANGEANIMATION); //Read Push Button
    FunctionMode = digitalRead(MODE_SWITCH);
    if ((PressedZ == LOW) and (ButtonAPress == false))
    {
        ButtonAPress = true;
    }
    TCNT1 = 0;    // Register initialisation
}

//Interrupts end Begin Main Program
void setup()
{
    strip.begin();
    strip.show(); // Initialize all pixels to 'off'
    pinMode(BUTTON_CHANGEANIMATION, INPUT_PULLUP);
    pinMode(SHOCK_SENSOR, INPUT_PULLUP);
    pinMode(MODE_SWITCH, INPUT_PULLUP);
    randomSeed(analogRead(0));
    noInterrupts(); // Disable all Interrupts
    TCCR1A = 0x00;
    TCCR1B = 0x02;
    TCNT1 = 0;    // Register initialisation
    OCR1A = 33353; // Load Output Compare Register
    TIMSK1 |= (1 << OCIE1A); // Activate Timer Compare Interrupt
    interrupts(); // Enable all Interrupts
}

//Helper Functions

void HSBToRGB
(
    unsigned int inHue, unsigned int inSaturation, unsigned int inBrightness,
    unsigned int *oR, unsigned int *oG, unsigned int *oB )
{
    if (inSaturation == 0)

```

```

{
    // achromatic (grey)
    *oR = *oG = *oB = inBrightness;
}
else
{
    unsigned int scaledHue = (inHue * 6);
    unsigned int sector = scaledHue >> 8; // sector 0 to 5 around the color wheel
    unsigned int offsetInSector = scaledHue - (sector << 8); // position within the
sector
    unsigned int p = (inBrightness * ( 255 - inSaturation )) >> 8;
    unsigned int q = (inBrightness * ( 255 - ((inSaturation * offsetInSector) >> 8) ))
>> 8;
    unsigned int t = (inBrightness * ( 255 - ((inSaturation * ( 255 - offsetInSector ))
>> 8) )) >> 8;
    switch( sector ) {
    case 0:
        *oR = inBrightness;
        *oG = t;
        *oB = p;
        break;
    case 1:
        *oR = q;
        *oG = inBrightness;
        *oB = p;
        break;
    case 2:
        *oR = p;
        *oG = inBrightness;
        *oB = t;
        break;
    case 3:
        *oR = p;
        *oG = q;
        *oB = inBrightness;
        break;
    case 4:
        *oR = t;
        *oG = p;
        *oB = inBrightness;
        break;
    default: // case 5:
        *oR = inBrightness;
        *oG = p;
        *oB = q;
        break;
    }
}
}

```

```

void CheckConfigButtons () // InterruptRoutine

```

```

{
bool PressedZ;
if (ButtonAPress == true)
{
    if (ShouldAnimation < MaxAnimationsAvail )
    {
        ShouldAnimation++;
    } else
    {
        ShouldAnimation = 0;
    }
    delay(400);
    ButtonAPress = false;
}
}

void AnimationControl ()
{
    int GetSelAnimation = 0;

    if (GetONOFFStatus != OLDDONOFFStatus)
    {
        OLDDONOFFStatus = GetONOFFStatus;
        if (GetONOFFStatus)
        {
            ShouldAnimation =1;
        } else
        {
            ShouldAnimation = 0;
        }
    }
}

// Main Loop -----

void loop()
{
    AnimationControl();
    RunAnimations();
    CheckConfigButtons();
}

// Main Loop ----- End
//Intros

void Intro_CountUp (byte r,byte g,byte b, int delaytime,bool dir)
{
    if (dir)
    {
        for ( int i = 0; i < strip.numPixels(); i++)
        {
            strip.setPixelColor(i,r, g, b);    //Calulate RGB Values for Pixel

```

```

        strip.show(); // Show results :)
        delay(delaytime);
    }
} else
{
    for ( int i = 0; i < strip.numPixels()+1; i++)
    {
        byte pos = strip.numPixels() - i;
        strip.setPixelColor(pos,r, g, b);    //Calculate RGB Values for Pixel
        strip.show(); // Show results :)
        delay(delaytime);
    }
}
}

void Intro_RaiseRainbow(bool risefall)
{
    brightness = 255;
    int Rainbowcolor = 0;
    if (risefall)
    {
        for (int i=0; i < strip.numPixels(); i++)
        {
            hue = map(i + Rainbowcolor, angleMin, 60, hueRedLow, hueRedHigh); //Set
Color
            HSBToRGB(hue, saturation, brightness, &r, &g, &b); //Set Color
            strip.setPixelColor(i, r, g, b);    //Calculate RGB Values for Pixel
            strip.show();
            delay(40);
        }
    } else
    {
        for (int i=0; i < strip.numPixels(); i++)
        {
            strip.setPixelColor(i, 0, 0, 0);
            strip.show();
            delay(40);
        }
    }
}

//Animations Outtros
void Ani_AllOff ()
{
    for ( int i = 0; i < strip.numPixels(); i++)
    {
        strip.setPixelColor(i,0, 0, 0);    // all off
    }
    strip.show();
}

```

```

void Ani_AllOn (byte r,byte g,byte b)
{
for ( int i = 0; i < strip.numPixels(); i++)
{
    strip.setPixelColor(i,r, g, b);    // all on
}
strip.show();
}

void Ani_Starshower ()
{
int array[10] ;
bool shockValue = true;
bool PressedT = true;

for ( int i = 0; i < strip.numPixels(); i++)
{
    strip.setPixelColor(i,0, 0, 15);    // all blue based
}
for (int i= 0; i< 10; i++)
{
    int selected = random(strip.numPixels());
    strip.setPixelColor(selected,255, 255, 255); // White
}
strip.show();
delay(100);
for ( int i = 0; i < strip.numPixels(); i++)
{
    strip.setPixelColor(i,0, 0, 15);    // all blue based
}
strip.show();
if(FunctionMode)
{
    delay(500);
} else
{
    do
    {
        shockValue = digitalRead(SHOCK_SENSOR);
        PressedT= digitalRead(BUTTON_CHANGEANIMATION);
        FunctionMode = digitalRead(MODE_SWITCH);
    } while ((shockValue)&&(!FunctionMode) &&( PressedT)) ;
}
}

void Ani_Rainbow(byte delaytime)
{
    brightness = 100;
    int Rainbowcolor = 0;
    bool shockValue = true;
    bool PressedT = true;

```



```

do
{
for (int i=0; i < strip.numPixels(); i++)
{
    hue = map(i + Rainbowcolor, angleMin, 60, hueRedLow, hueRedHigh);
    HSBToRGB(hue, saturation, brightness, &r, &g, &b);
    strip.setPixelColor(i, r, g, b);
}
strip.show(); // Show results :)
if(FunctionMode)
{
    delay(delaytime);
} else
{
    do
    {
        shockValue = digitalRead(SHOCK_SENSOR);
        PressedT= digitalRead(BUTTON_CHANGEANIMATION);
        FunctionMode = digitalRead(MODE_SWITCH);
    } while ((shockValue)&&(!FunctionMode) &&( PressedT)) ;
}
Rainbowcolor++;
} while (Rainbowcolor < 61);
}

void Ani_Two_Color ()
{
bool shockValue = true;
bool PressedT = true;
byte Divider = random (1,10);
bool color;
int x = 1;
b = 0;
for (int s = 0; s > -1; s = s + x)
{
    color = false;
    for ( int i = 0; i < strip.numPixels(); i++)
    {
        a = i / Divider;
        if (!(a == b))
        {
            b = a;
            color = !color;
        }
        if (color) { strip.setPixelColor(i, 0, s, 0); } //grün
        if (!(color)) { strip.setPixelColor(i, s, 0, 0); } //rot
    }
    strip.show();
    if (s == 255)
    {

```

```

    if(FunctionMode)
    {
        x = -1;
        delay(200);
    } else
    {
        do
        {
            shockValue = digitalRead(SHOCK_SENSOR);
            PressedT= digitalRead(BUTTON_CHANGEANIMATION);
            FunctionMode = digitalRead(MODE_SWITCH);
        } while ((shockValue)&&(!FunctionMode) &&( PressedT)) ;
        x = -1;
    }
    delay(10);
}
strip.show();
}

void Ani_Halloween()
{
    bool shockValue = true;
    bool PressedT = true;
    a = -10;
    for (int i=0; i < strip.numPixels(); i++)
    {
        strip.setPixelColor(i, random(1,254), random(1,204), random(1,254));
        e = e + a;
        f = f + a;
        if (f <= 0)
        {
            a = +10;
        }
        if (f >= 60)
        {
            a = -10;
        }
    }
    strip.show(); // Show results :)
    if(FunctionMode)
    {
        delay(300);
    } else
    {
        do
        {
            shockValue = digitalRead(SHOCK_SENSOR);
            PressedT= digitalRead(BUTTON_CHANGEANIMATION);
            FunctionMode = digitalRead(MODE_SWITCH);
        } while ((shockValue)&&(!FunctionMode) &&( PressedT)) ;
    }
}

```

```

    }
}

void FadeColor ()
{
    byte brightness = 0;
    byte saturation = 0;
    int Colori = 49 ;
    do
    {
        for (int i=0; i < strip.numPixels(); i++)
        {
            HSBToRGB(Colori, saturation, brightness, &r, &g, &b); //Set Color
            strip.setPixelColor(i, r, g, b); //Calculate RGB Values for Pixel
        }
        brightness ++;
        strip.show(); // Show results :)
        delay(40);
    } while (brightness < 50);
}

void RunAnimations()
{
    if (!(ShouldAnimation == IsAnimation))
    {
        PlayOutro = true;
        ChangeAnimation = true;
    }
    switch (IsAnimation)
    {
        case 0: // all LedsOFF
            if (PlayIntro)
            {
                PlayIntro = false;
                RunOnce = true;
            }
            if ((!(PlayIntro)) && (!(PlayOutro)))
            {
                if (RunOnce) { Ani_AllOff (); }
                RunOnce = false;
            }
            if (PlayOutro)
            {
                PlayOutro = false;
                PlayIntro = true;
                RunOnce = true;
                IsAnimation = ShouldAnimation;
            }
            break;
        case 1:
    
```

```

    if (PlayIntro)
    {
        Intro_CountUp (0,0,15,100,true);
        PlayIntro = false;
    }
    if ((!(PlayIntro)) && !(PlayOutro)))
    {
        Ani_Starshower();
    }
    if (PlayOutro)
    {
        Intro_CountUp (0,0,0,100,false);
        PlayOutro = false;
        PlayIntro = true;
        IsAnimation = ShouldAnimation;
    }
    break;
case 2:
    if (PlayIntro)
    {
        Intro_RaiseRainbow(true);
        PlayIntro = false;
    }
    if ((!(PlayIntro)) && !(PlayOutro)))
    {
        Ani_Rainbow(20);
    }
    if (PlayOutro)
    {
        Intro_RaiseRainbow(false);
        PlayOutro = false;
        PlayIntro = true;
        IsAnimation = ShouldAnimation;
    }
    break;
case 3:
    if (PlayIntro)
    {
        Ani_Alloff ();
        PlayIntro = false;
    }
    if ((!(PlayIntro)) && !(PlayOutro)))
    {
        Ani_Two_Color (); // Ani_Two_Color (byte hue,byte tail,byte brightness,byte
delaytime)
    }
    if (PlayOutro)
    {
        PlayOutro = false;
        PlayIntro = true;
        IsAnimation = ShouldAnimation;

```

```

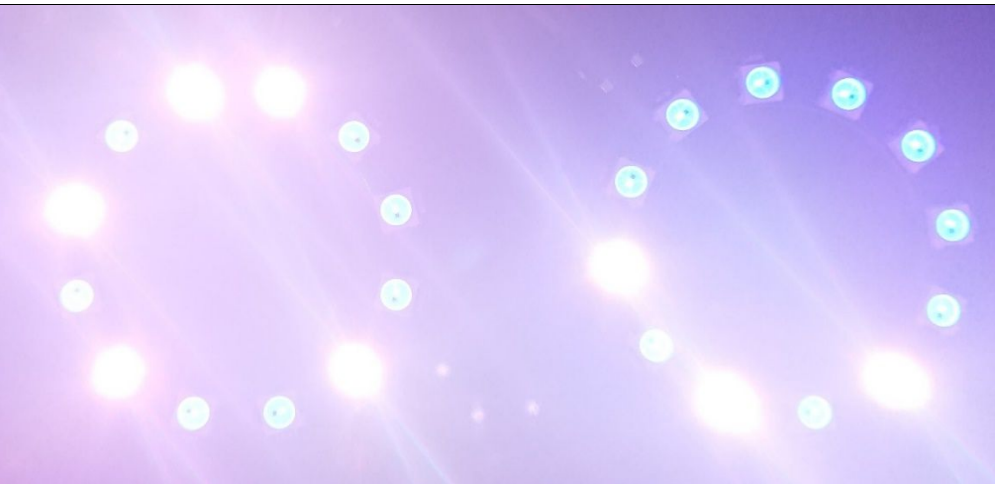

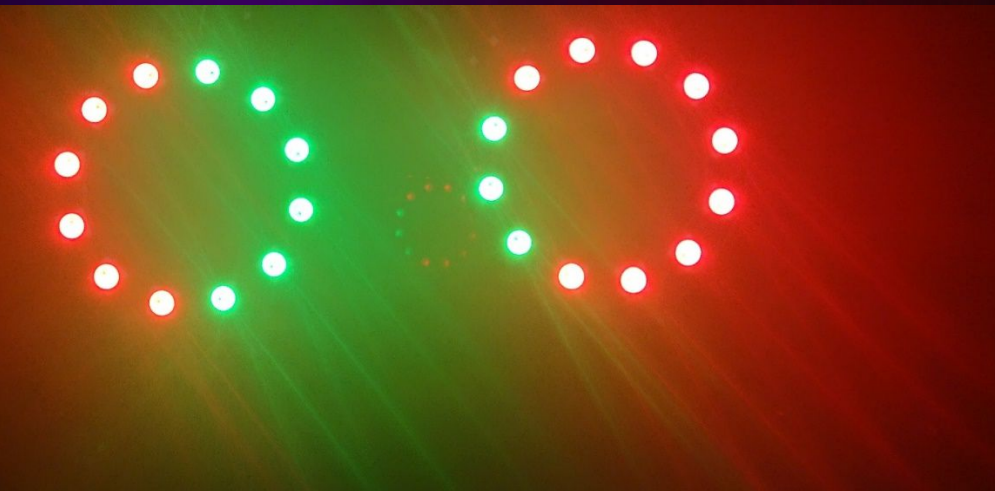
    }
    break;
case 4:
    if (PlayIntro)
    {
        Ani_AllOff ();
        PlayIntro = false;
    }
    if (!(PlayIntro)) && !(PlayOutro))
    {
        Ani_Halloween (); //
    }
    if (PlayOutro)
    {
        PlayOutro = false;
        PlayIntro = true;
        IsAnimation = ShouldAnimation;
    }
    break;
}
}

```

Durch kurzen Druck auf den Taster können die Animationen nun auch wieder nacheinander aufgerufen werden. Neu ist, das jetzt durch einfaches betätigen des Schalters zwischen den Funktionsmodis „Standalone“ und „Feedback“ umgeschaltet werden kann.

Es reagiert dabei jede! der insgesamt vier Animationen leicht unterschiedlich auf den Vibrationssensor, falls der Modus „Feedback“ gewählt wurde.

Als kleine Vorstellungshilfe habe ich einmal alle 4 unterschiedlichen Animationen fotografiert:

<p>Starshowe r</p>	 <p>A light pattern on a purple background featuring two concentric circles of lights. The inner circle consists of 12 bright white lights, and the outer circle consists of 12 cyan lights. A small cluster of 5 white lights is located in the center.</p>
<p>Rainbow</p>	 <p>A light pattern on a dark background featuring two concentric circles of lights. The inner circle consists of 12 magenta lights, and the outer circle consists of 12 yellow-orange lights. A small cluster of 5 white lights is located in the center.</p>
<p>Motion of Two Colors</p>	 <p>A light pattern on a dark background featuring two concentric circles of lights. The inner circle consists of 12 green lights, and the outer circle consists of 12 red lights. A small cluster of 5 white lights is located in the center.</p>

Halloween



Bitte beachtet, das in beiden! Modis die Animationen nicht sofort auf die nächste Animation umschalten, sondern immer erst die aktuelle Sequenz beenden, bevor die nächste Animation gestartet wird.

Ich wünsche nun viel Spaß beim Nachbauen der Discobrille. Vielleicht möchtest du ja weitere Animationssequenzen dazu programmieren, falls dir die 4 vorhandenen nicht ausreichen sollten?

Probiere auch mal andere Sensoren aus, wie zb:

- [SW420 Vibrations und Schüttel Sensor](#)
- [Schock Sensor Modul](#)
- [KY-020 Shake/Shock Sensor Modul](#)

Oder Ihr versucht es mit einem Sensor aus den verschiedenen

- [Sensoren aus dem Sensorkit ?](#)

Schreibt mir eure Ideen oder Fragen in die Kommentare.