

Robotikpraktikum - POV Globus

Max Hartmann, Philipp Gernandt & Tobias Buck
(Physik)

Betreuer: Gero Plettenberg & Thomas Kloepfer

06.10.2015

- 1 Projektziel
- 2 POV-Prinzip
- 3 Komponenten
- 4 Realisierung und Code
- 5 Probleme
- 6 Ausblick

Projektziel

Zielvorgaben

- Konstruktion eines POV Globus
 - Stabile Anzeige ohne Walk-Off
 - Einfaches Einlesen/Anzeigen von Bildern

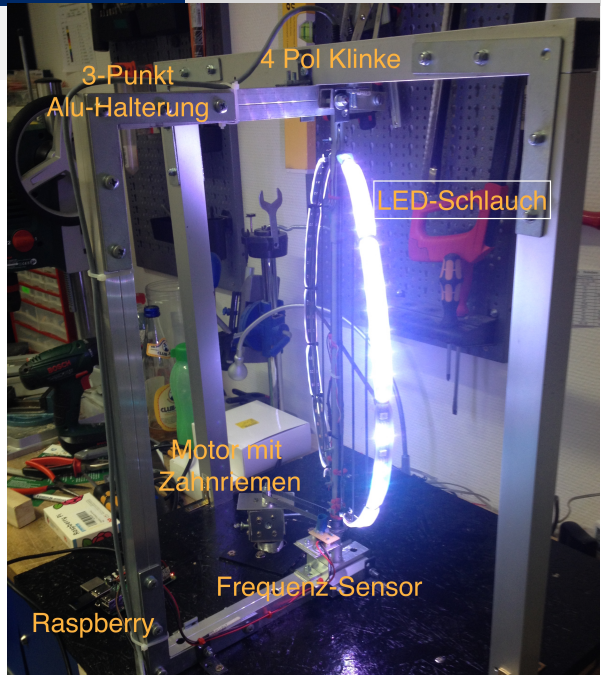
Umsetzung

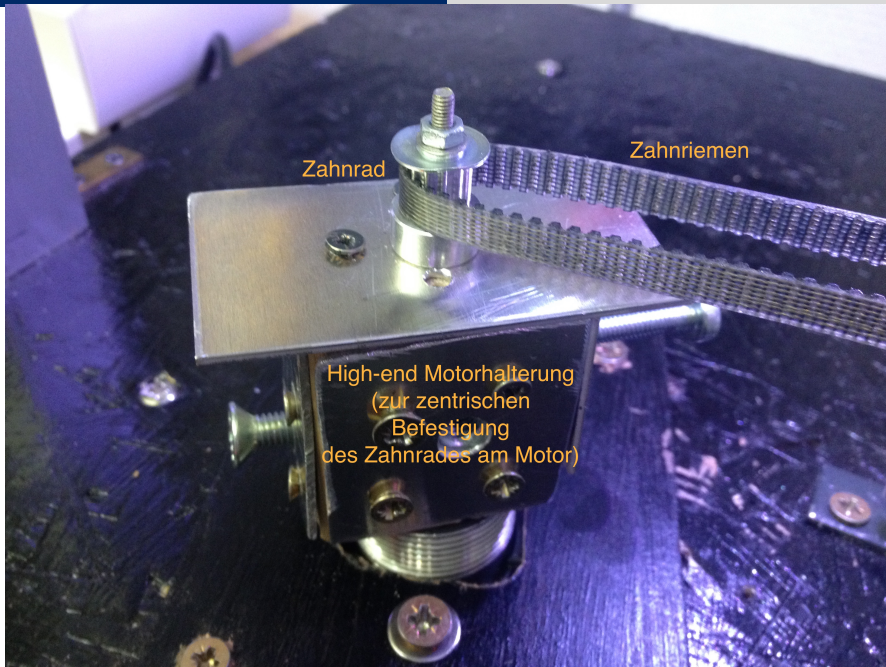
- Steuerung durch einen Raspberry Pi
- Adafruit Dotstar LED Strip (60 RGB-LEDs; ca. 4kHz)
- Standventilatormotor als Antrieb (ca. 7 Umdrehungen/s)
- Programmierung in Python

Persistence Of Vision (POV)

- Das Auge kann Reize nur mit einer bestimmten Frequenz verarbeiten (z.B. Filme: 24Hz; Cartoons: 12Hz)
 - ⇒ schnell bewegte Lichtquelle erscheint als Linie
 - ⇒ schnell bewegte, *blinkende* Lichtquelle erscheint als mehrere Pixel
- Schnell rotierender Kreis ⇒ Kugeloberfläche
- + (synchron zur Drehfrequenz) blinkende LED-Zeile ⇒ POV-Globus

Komponenten

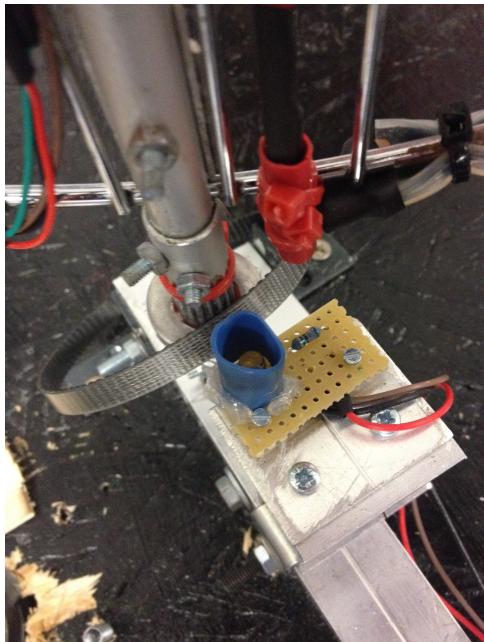
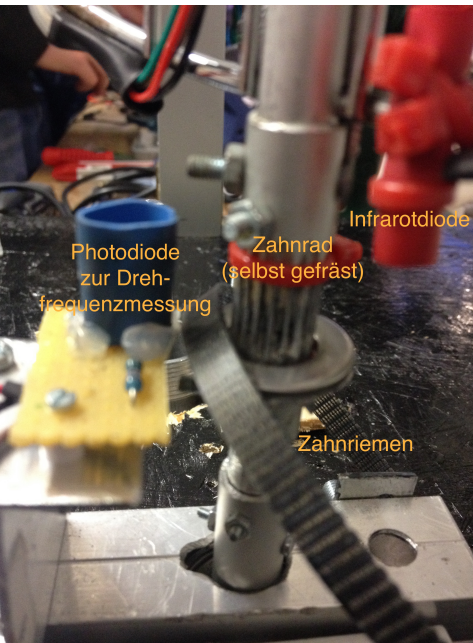


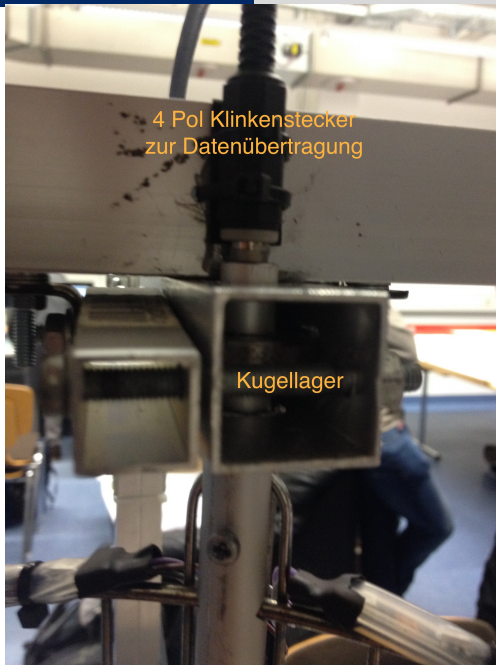




Ventilatormotor

Motorgeschwindigkeitsregler





Features



Datenübertragung über 4 Pol Klinke (axial gelagert)

- Idee: Nur der LED-Schlauch soll rotieren
- LED Schlauch benötigt 4 Anschlüsse: VCC, GND, DATA, CLOCK
- Übertragung der Daten auf rotierende Elemente via 4 Pol Klinke

Datenaufbereitung

Zur optimalen Ausnutzung des 60 LED-Schlauchs:
Invertierung und Schachtelung der Zeilen!

Mapping:

LED 0 → Zeile 58

LED 30 → Zeile 1

LED 1 → Zeile 56

LED 31 → Zeile 3

LED 2 → Zeile 54

LED 32 → Zeile 5

...

...

LED 29 → Zeile 0

LED 59 → Zeile 59

⇒ Formel:

LEDs 0 - 29: $i \rightarrow 60 - 2 \times (1 + i)$

LEDs 30-59: $i \rightarrow 1 + 2 \times (i - 30)$

Datenaufbereitung (Python)

```
# Load image in RGB format and get dimensions:
img          = Image.open(inFile).convert("RGB")
pixels       = img.load()

# create empty array:
nPixels=[0 for i in range(nwidth)]
for col in range(nwidth):
    nPixels[col]=[(0,0,0) for line in range(60)]
pixels_temp=nPixels # copy for work...

# re-combine lines in order to use second part of strip correctly
for cLine in range(height/2): # copy image and reorder lines
    for col in range(width):
        pixels_temp[col][29-cLine] = pixels[col, 2*cLine] # even
        pixels_temp[col][30+cLine] = pixels[col, 2*cLine+1] # odd
if height%2==1: # last line
    for col in range(width):
        pixels_temp[col][29-(height/2)] = pixels[col, 2*(height/2)]
```


Datenaufbereitung (Python)

Außerdem: Verschiebung der Spalten der ungeraden Zeilen!

- gerade Zeilen: unverändert.
- ungerade Zeilen:
 - Spalte 1-29 → Spalte 30-59
 - Spalte 31-59 → Spalte 0-29

```
for parts in range(nwidth/60): # reorder odd lines
    for col in range(30):
        for cLine in range(30):
            cCol=col+parts*60 # current column

            nPixels[cCol][cLine]=pixels_temp[cCol][cLine] # even
            nPixels[cCol+30][cLine] = pixels_temp[cCol+30][cLine]

            temppix=pixels_temp[cCol][cLine+30] # odd
            nPixels[cCol][cLine+30] = pixels_temp[cCol+30][cLine+30]
            nPixels[cCol+30][cLine+30] = temppix
```

Ansteuerung und Datenübertragung

Jede LED benötigt 4 bytes: (0, blau, grün, rot)

⇒ Schlauch wird durch Liste von $4 \times 60 = 240$ bytes gesteuert.

Die Umsetzung wird durch die Bibliothek *dotstart* von *Adafruit_DotStar* mittels Clock- und Data-PINs (GPIO) realisiert.

Ansteuerung und Datenübertragung

Wichtig: Exakt richtiges Timing!

~ 7 Umdrehungen pro Sekunde = 60 Spalten $\Rightarrow \sim 0.0024\text{s}$ pro Spalte
 \rightarrow kleine Abweichung führt zu Verschiebung im Bild!

Lösung zur Stabilisierung:

- Infrarot-LED misst Rotationsfrequenz
- Script überprüft verstrichene Zeit im Code
- \Rightarrow angepasste Verzögerung zwischen einzelnen Spalten

Datenübertragung (Python)

```
# initialize values
pos=0
timeB=time.time()
timeDiff=0
timeDiff_new=0

while True: # Loop until KeyboardInterrupt

    period=freq.period
    pixel_time=(period-timeDiff)/60.

    for x in range(60):                # For each column of image...
        strip.show(array[x+pos*60])    # Write raw data to strip
        time.sleep(pixel_time)

    pos=(pos+1)%(width/60)             # next slice/rotation
    actPeriod=time.time()-timeB
    timeDiff_new=actPeriod-period
    timeDiff=(timeDiff+timeDiff_new)%period
    timeB=time.time()
```

Probleme

- Probleme mit diversen Motoren
→ diverse Konzepte ausprobiert
- Datenübertragung (Abrieb des Klinkensteckers; fehleranfällig)
- Lagerung des Kreises
- zentrische Lagerung der gesamten Konstruktion
- exakte Anpassung des Skripts an Rotationsfrequenz
- zerstörter LED-Schlauch

Ausblick

- Verbesserung der Bildverarbeitung
 - bewegte Bilder
 - Schrift / Laufschrift
 - Spiele programmieren
- bessere Lagerung
- besserer Motor
- höhere Anzahl an Pixel

Vielen Dank für Ihre Aufmerksamkeit