Spinning LED Design

© 2023, Joris Dobbelsteen. All rights reserved.

# Objective

Rotating bar of LEDs to project an image all around. The image can be a rotating text, rotating image or image captured by the camera (low frame rate acceptable).

Rotation speed should be high enough (expected above 15 updates/sec) so persistence of vision works. Higher update rates are preferred.

The device should be powered externally (not though a battery). This requires power transfer between stable and moving world.

## Concept

LEDs

Motor

Camera

Controller

LEDs

Power & Data  
transfer

Control Panel

# Design concerns

* Image resolution
  + Update LEDs fast enough to project an image
* Power ~~& (especially) data transfer~~ from “fixed” to “moving” world

## Safety concerns

* If the device is large enough, it will spin a bar at high speed. Injury due to mechanical failure or touching the bar
* Imbalance of rotating part can lead to failure
* User control must be outside moving parts

# Design Choices

* Parts will be sourced as-is or 3D printed. Can source PCB & electronic components for critical parts (e.g. moving world)
  + Printed parts must fit on 30x30 bed of Ender 3 S1 Plus
* Build around HD107S RGB leds, 144 leds/meter (fast & easy to control)
  + SPI like interface up to 40 MHz
  + Refresh rate of 27 kHz, but can we update that fast?
  + These seem to be the highest speed led strips that can be purchased at the moment.
* STM32 is preferred controller (familiarity & tools available). Other options are possible if they make more sense
* Build around BLDC motor (since I have an STM32 motor controller board). Or can create an updated one as a learning exercise. Motor is on the “fixed” world due to power requirements.
  + Motor selected is Dys D2830 750KV (7.4 to 14.8V, up to 185W)
* ~~Need a camera sensor, but update rate can be very low (~1 fps). Camera must be on fixed world due to motion blur~~
* Camera & mode control will be on an Android tablet.  
  Apple will not be supported due to cost for development environment.
* The device should be enclosed in a transparent box while the arms are moving. Due to the high speed of the arms, there is a risk of injury when they come in contact with a person or object. Or when the arms fail due to mechanical stress.

# Physical Design

The design has several main concerns:

* Size of arm & motor RPM with led bar, due to travelling speed
* Stability of moving part
* …

## Major Physical Design criteria

|  |  |  |
| --- | --- | --- |
| Metric | Value | Rationale |
| Arm Radius | 22 cm | Max 30 cm. Casing limits diameter to below 50 cm. |
| Arm Height | 40 cm | Casing limits to below 50 cm. 40 cm retains clearance. |
|  |  |  |

## LED Arm Design

The arm with LEDs is spinning at a fast speed.

### Centrifugal force

Formula:

F = m*ω2r*

First we need a weight estimate:

* LED strips consume up to 40 W/m. At 40 cm, that is 16W or 3.2A@5V. At least 0.2mm2 wire. We calculate with 0.5mm2, which is <2g/m.
* For PLA, we estimate 1.25 gr/cm3.

|  |  |
| --- | --- |
| Object | Weight (g) |
| LED strip (40cm) @ 80g/m | 36 |
| Wiring | 2 |
| Arm (40x1,5x0,25 cm) | 20 |
| Supports (5x 50x1,5x0,3cm) | 150 |
|  | 208 |

See “Spinning LED Design Calculations.xlsx” for details.

Assuming that:

* The arm support is solid PLA (of 225mm2 of connecting material)
* Forces are applied in XY printing direction (not aimed at breaking apart printed layers)

This design should be able support arm with 50 cm distance and 3000 rpm (50 rot/sec).

Recommended:

* up to 30 cm radius
* Arm supports at least 225 mm2 (15x15 mm) pure PLA supports (margin of safety = 1)

### LED Arm speed

See “Centrifugal Force & Speed” sheet in Excel document. Main results are:

|  |  |  |  |
| --- | --- | --- | --- |
| Arm Radius | Speed 20 rps / 1200 rpm | Speed 25 rps / 1500 rpm | Speed rps 30/ 1800 rpm |
| 15 cm | 19 m/s / 68 km/h | 24 m/s / 85 km/h | 28 m/s / 102 km/h |
| 20 cm | 25 m/s / 90 km/h | 31 m/s / 113 km/h | 38 m/s / 136 km/h |
| 25 cm | 31 m/s / 113 km/h | 39 m/s / 141 km/h | 47 m/s / 170 km/h |
| 30 cm | 38 m/s / 136 km/h | 47 m/s / 170 km/h | 57 m/s / 204 km/h |

Concern: what is a a safe speed?

If we put a casing around this, a typical size for acrylic/plexiglass is 50x50 cm sheets. So that limits the arm radius to 22 cm to give clearance.

### Drag resistance

Moving at an estimated 100 km/h (or faster), the drag force is significant. This needs to be overcome by the motor.

FD = ½ ρ v2 CD A

where

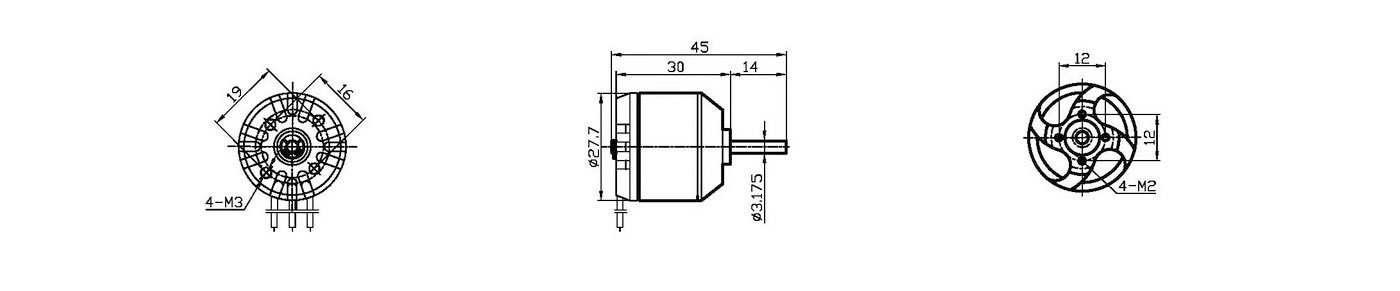
* FD is the drag force,
* ρ is the [density](https://en.wikipedia.org/wiki/Density) of the fluid (constant in this case),
* v is the speed of the object relative to the fluid,
* A is the [cross sectional area](https://en.wikipedia.org/wiki/Cross_section_(geometry)), and
* CD is the [drag coefficient](https://en.wikipedia.org/wiki/Drag_coefficient) – a [dimensionless number](https://en.wikipedia.org/wiki/Dimensionless_number).

A trade-off has to be found that balances these parameters:

* Speed (v) is quadratic element (having a large impact) is impacted by:
  + Requirement to maintain pleasant Persistence of Vision (POV) with a high enough frame rate. The number of arms (either 2 or 4) influences the rotation speed (theoretically by a factor of 2).
  + Choosing interlaced images influences the rotation speed (theoretically also by a factor of 2).
  + The speed increases linearly with the length of the arm (distance of LEDs from center rotation line of the device).
* The shape of the arm impacts the surface area (A) and shape which determines the drag coefficient (CD). This has to be balanced with the required stiffness to give a standstill image as well as
  + Having the arm exert a lift force towards the center point, counteracting the centrifugal forces, might result in a more optimized arm shape.

## Motor & Axis Mount Design

### Design Concept



Base plate of “fixed world”.

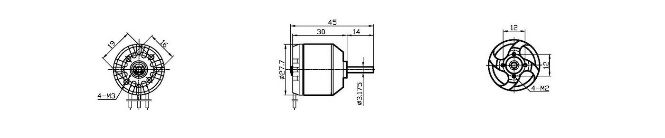
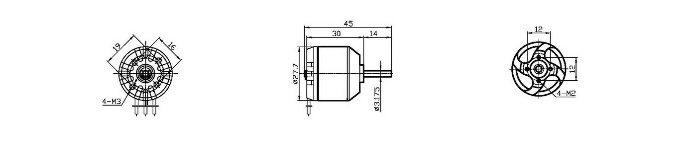
Motor coupler.  
Or: use holes in motor outrunner.

Bearings (motor might help as well).

Contacts (power & data)

“Moving world”

Mount.  
Can fit smaller motors.

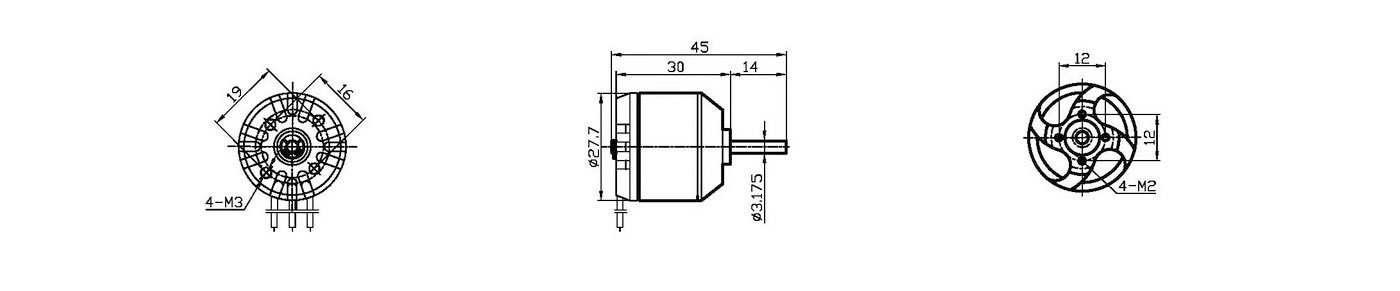


Brushes or springs

Detection of rotation

### Motor dimensions

<http://www.dys.hk/product/D2830.html>



# LED strip

Choice:

* Interlaced due to visual quality
* Arms can be variable

## Physical appearance

|  |  |  |
| --- | --- | --- |
| Metric | Non-interlaced / Interlaced | 2 arm / 4 arm |
| Visual | Interlaced has no gap between pixels Smaller pixels (double resolution) | Not affected |
| Rotation speed | Half / Full speed (with equal number of arms) Non-interlaced can use that both arms show same image. | Full / Half speed 2nd set of arms duplicates the image, allowing reduced speed. |
| LED update speed | Quarter / Full speed (with equal arms) Non-interlaces has half resolution to keep pixels square. This is in addition to requiring a low rotation speed. | From total pixels per second point, same update rate for both cases. Either double speed and half the arms or half the speed and double the arms. For each LED strip on an arm, double arms lower the update per second required. |
| Memory | Quadruple for interlaced to keep pixels square. Meaning, for interlace both vertical as well as horizonal resolution double. | Not affected |
| MCU Peripherals | Not affected | Total (summed) SPI bitrate is equal. 4 arms requires either:   * 2x SPI with long connecting bus * 2x SPI, switching output between led strips * 4x SPI (can be at half speed) |

* Non-interlaced (each arm shows same data) vs interlaced (each arm shows alternating line)  
  No point beyond interlaces with 144 pixels/m and 5050 chips.
* 2 arm requires 20 rotations per second.  
  4 arm requires 10 rotations per second.

## Rotation & update speed

See “LED Toggle Rate” sheet in Excel document.

# System Design

## Working concept

The design relies on:

* Data communication to the On-Axis part happens via wireless protocol, for which WiFi is selected due to bandwidth for the camera. Only power is transferred via physical connection.  
  This greatly simplifies the mechanical/electrical system, removing the need for high-speed (megabit/second) data transfer over a rotating coupling.
* Active part of the rotation detection is on the On-Axis part.
* Rotation sensor on on-mount is optional and should be used solely for motor control.

On-Axis (Moving World)

Axis MCU

LED Strips

Rotation Sensor

On-Mount (Fixed World)

ESC

Rotation Sensor

Motor

Mobile

Tablet with  
App

Camera

WiFi  
MCU

## Concept v0.1

Below concept is rejected because:

* Slip ring is highly complicated due to requirement of high-speed communication. Simplifying this to only power (and maybe very low-speed communication if absolutely needed) is much more reliable to achieve.
* On-mount part can be simplified

On-Axis (Moving World)

Axis MCU

LED Strips

Rotation Sensor

On-Mount (Fixed World)

Mount MCU

Camera

SD Card

USB

Network

Touch Screen

Rotation Sensor

Motor

# ESC Application

Any BLDC Electronic Speed Controller that can keep a fixed speed of up to 1000 rpm (depending on axis/LED configuration)

* Off-the-shelf ESC  
  plus PWM controller
* ST Microelectronics B-G431B-ESC1
* Integrated custom design (e.g. based on B-G431B-ESC1)

# Axis MCU Application

LED Strip Output

Image reception

Rotation  
trigger

Data IF

Image  
Buffer1

LEDS SPI1

LEDS SPI2

Image Buffer Status

pixels

pixels

Pixel Line Buffer

Format conversion

Timing Control

DMA

## Requirements and CPU selection

|  |  |  |
| --- | --- | --- |
| Criteria | Requirement | Description |
| Image buffer | 91 KB | 22cm radius, 40 cm high, 30 rot/sec, interlaced 🡪  400 \* 116 \* RGB565 (16-bit) |
| ~~Image buffer extra lines for motion~~ | ~~??? 15 KB~~ | ~~Update rate is 4 Hz. Hence 1/6~~~~th~~ ~~of the screen can be updated. But scroll rate is better estimate.~~ |
| Pixel Line buffer | 264 B or 528 B | 2x ~ 4x SPI double-buffered 58 pixel buffer 2~4 \* 2 \* (58 \* RGBX8888 (32-bit) + 2 \* 32-bit) |
| Reception buffer | 256 B | 116 px \* 24 cycles 🡪 ~45 microseconds @ 64 MHz In this period:   * ~36 bytes received at 8 MBaud for UART * 90~ received at 20 Mbps for SPI   Can simplify code by buffering entire pixel line + 18 byte |
| Stack | 1 KB |  |
| Other data | ??? 2 KB |  |
| Total | 112 KB 96 KB | With extra motion buffer (estimate) Without extra motion buffer |

## Inputs

|  |  |  |
| --- | --- | --- |
| Input | Type | Description |
| Rotation Trigger | Pulse | Edge-trigger for rotation detection (index) |
| Data IF | U(S)ART or SPI | Commands, images & text data. Needs parity. |
|  |  |  |

## Outputs

|  |  |  |
| --- | --- | --- |
| Input | Type | Description |
| LEDS | 2x SPI | Pixel data to HD107s Need other IC for 4x SPI |
| LEDS OE | 2x ~ 4x GPIO | Output enable for pixel data in case of 4 arms There are no 4 SPIs available |
| Status LED | GPIO (or timer) | Single led |

## CPU Selection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CPU | CPU | RAM | SPI | RGB565 conv. | Price (10pc/Mouser) |
| STM32G0B*x* | CM0+ 64 MHz | 144KB | 3 @ 32 Mbps | CPU | € 4 |
| ~~STM32F401xD STM32F401xE~~ | ~~CM4 84 MHz~~ | ~~96KB~~ | ~~4 @ 42 Mbps~~ | ~~CPU~~ | ~~€ 6~~ |
| STM32F411 | CM4 100 MHz | 128KB | 3 @ 50 Mbps 2 @ 25 Mbps | CPU | € 7 |
| STM32F412 | CM4 100 MHz | 256KB | 3 @ 50 Mbps 2 @ 25 Mbps | CPU | € 8 |
| STM32F413 | CM4 100 MHz | 320KB | 3 @ 50 Mbps 2 @ 25 Mbps | CPU | € 10 |
| STM32F427 STM32F429 | CM4 180 MHz | 256KB | 4 @ 45 Mbps 2 @ 22.5Mbps | ChromArt | € 14 |
| STM32G473M/V/Q | CM4 170 MHz | 128KB\* | 4 @ 75 Mbps | CPU | € 9 |
| STM32G474M/V/P/Q | CM4 170 MHz | 128KB\* | 4 @ 75 Mbps | CPU | € 9 |

## Performance on STM32G0B*x* (Cortex-M0+)

As we might need around 100 KB just for pixel data, the STM32G0B0 series is the lowest cost CPU that suits or purposes. It features:

* Cortex-M0+ CPU and bus at 64 MHz
* DMA to offload CPU
* 144 KB SRAM (or 128KB with parity)

Worst case:  
30cm radius, 50 cm high, 30 rot/sec, interlaced pixels = 16286 columns/sec 🡪  
*columns/sec* \* 144 pixel in height = 2345184 px/sec

|  |  |  |
| --- | --- | --- |
| Process | % | Description |
| Image Reception |  | DMA required with periodic check |
| Timing Control | 4% | 16286 columns/sec Interrupt latency + exit: 2 \* (15 + 2WS) = 34 cycles Routine estimated at 110 cycles |
| Format Conversion | Up to 90% | 64 MHz / 2345184 px/sec 🡪 Up to 27 cycles/pixel for conversion Estimate 24 cycles/pixel *At 20 cycles/px this takes up to 75% CPU* |

Realistic case:  
22cm radius, 40 cm high, 30 rot/sec, interlaced pixels = 11944 columns/sec 🡪  
*columns/sec* \* 116 pixel in height = 1385504 px/sec 🡪 ~1400000 px/sec

|  |  |  |
| --- | --- | --- |
| Process | % | Description |
| Image Reception |  | DMA required with periodic check |
| Timing Control | 3% | 11944 columns/sec Interrupt latency + exit: 2 \* (15 + 2WS) = 34 cycles Routine estimated at 110 cycles |
| Format Conversion | Up to 55% | 64 MHz / 1400000 px/sec 🡪 Up to 45 cycles/pixel for conversion Estimate 24 cycles/pixel |

### Bus load

Worst-case design, assume bus running at maximum 64 MHz clock.

|  |  |  |  |
| --- | --- | --- | --- |
| Process | SRAM % | APB % | Description |
| Image Reception – UART | 0.7% | 0.7% | 4MBaud 🡪 400.000 bytes/sec |
| Image Reception – Mem2Mem | 1.3% |  | 400.000 bytes copy inside SRAM |
| LED Strips – 2x SPI | 6.5% | 6.5% | SPI runs at 50% clock rate and DMA transfer every 16 bits 🡪 2 DMA transfers / 32 cycles. Dead-time lowers load. *Double load for 8-bit DMA transfers.* |
| CPU reservation | < 20% |  | Assume one in every 5 cycles = every 3 instructions |

### RGB565 to RGBX8888 pixel conversion algorithms (Cortex-M0+)

In order to keep the RAM of the device limited and enable selecting a low-cost CPU, it’s important to use RGB656 (16-bit) storage format. Since LED strips require RGBX8888, pixels data must be converted.

At 24 cycles/pixel we would load CPU up to 90%. Hence this is an area that needs special attention and ensure the cycles budget is met.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | Cycles/px | Stalls/px | Flash | RAM | Description |
| Lookup 16-bit (flash) | 14 | ~1 | 256KB |  | Full table in Flash (64K\*4-byte) |
| Lookup 8-bit (flash) | 23 | ~1 | 1KB |  | Lookup RGB565 in two steps (2 \* 256\*2-byte) |
| Lookup 8-bit (SRAM) | 19 | ~2 |  | 1KB | Compute at boot or as “data” (2 \* 256\*2-byte) |
| Lookup 8-bit (flash) | 22 | ~1 | 1.5KB |  | Lookup RGB565 in two steps (256\*2-byte + 256\*4byte) |
| Lookup 8-bit (SRAM) | 18 | ~2 |  | 1.5KB | Compute at boot or as “data” (256\*2-byte + 256\*4byte) |
| Compute | 19 | ~1 |  |  | Imprecise (last color bits always 0) |
| Compute per 2 px | ? | ? |  |  | Load 2x 16-bit RGB565 as 32-bit value |

For worst-case design: At the cost of 256KB flash, CPU consumption can be lowered to 60%. This makes the trade-off worthwhile. Using 1.5 KB RAM would lower the CPU consumption to 80%.

For the realistic design the algorithm should be kept <= 32 cycles to give ample time for other activities.

## Performance on STM32 with Cortex-M4

These Cortex-M4 CPUs should fit easily if the Cortex-M0+ is appropriate. Hence no analysis is done since:

* The Cortex-M4 is more efficient than Cortex-M0+. This is mainly due to instruction set and the device having an instruction and data bus, rather than a single shared bus.
* The Cortex-M4 designs run at a higher clock frequency.