Persistence of Vision Display

Arthur Deleu Tevon Eyabi Jannes Op de Beeck
KU Leuven Campus De Nayer
Sint-Katelijne-Waver

Abstract

This article describes the design process of a POV display. Our main goal was to create a user friendly POV display that can display high quality images. We also created an application alongside the display. It goes over possible options to design this project, and explain the choices we made. We will go into detail about the different parts of the hardware and software, and explain the steps we took to make it. Then we will talk about some of the problems we encountered, analyse our eventual result and add suggestions for future revisions.

1. Introduction

An important task for universities is getting new students interested in joining them. This can be done by having impressive projects to show. Therefore, KU Leuven Campus De Nayer wanted a persistence of vision display being made.

Persistence of vision (POV) is an optical phenomenon which makes that humans perceive an image for longer than the existance of the actual image. Thanks to this phenomenon, a single line of LEDs that turns around at a high enough speed can be used as a display.

This project aims to create a POV display for the university, with high image quality and a user friendly interface, by developing both the software and hardware for it.

2. Related works

To start this project, we took a look at existing POV display projects, to research the effects of different ways of designing the displays. There were 2 different types of POV displays we found while looking up existing projects, with the LEDs either perpendicular to the axis of rotation, which creates a circle, or parallel with it, creating a cylinder [1]. Another difference we saw between projects was the way to power the display, and transfer the data. Sometimes, this was done using slip rings [3], or using a battery and wireless communication [2].

3. Approach

To start, we opted to make a circular display, since the visible area is much larger, and flat. As a result, the image quality will be better. Next provide the display with power, we decided on using a battery. This eliminates the noise and wear caused by the brushes of a slip ring.

3.1. Hardware

3.1.1 Choice of components

The first and most important choice we had to make was the type of LEDs. The LEDs need to be easily controllable, able to refresh quickly, and capable of displaying RGB colors. With these properties in mind, two logical choices emerged: the APA102C or the SK9822-EC20. The major difference between these two options is their size. The APA102C has a width of 5mm, while the SK9822-EC20 has a width of 2mm. With smaller leds, the display can have a higher resolution, which is why we chose the SK9822-EC20.

The next most important component was the microcontroller. We aimed to transmit our images wirelessly, so the microcontroller must be capable of easily connecting with other devices. Additionaly, the microcontroller needs to possess a significant amount of memory to store the images. Lastly, to be able to precisely switch a large number of LEDs at high frequency, it needs to be fast. With these requirements in mind, we decided on the ESP32-S3-MINI-1U-N8. With a clock frequency of 400 MHz, 39 GPIO pins, on-board 2.4 GHz Wi-Fi, and 512 KB SRAM, this microcontroller satisfied all our needs.

3.1.2 Design of the PCB

F We had to design our own PCB, because the components we chose were all SMD components. Before the PCB could be made, we needed to draw the complete electrical circuit of our display. When this was done, we had to decide the

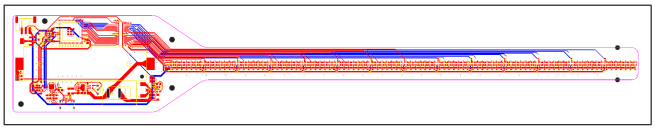

Figure 1. The final PCB design.

physical dimensions of our PCB, and place the components. The final design is shown in figure 1

3.2. Software

3.2.1 Image encoding

To display the image on the display, we had to create a function that transforms it in multiple ways. First we had to change the coordinate system of the image from cartesian to pole coordinates by using the following formulas on every pixel:

$$r = \sqrt{x^2 + y^2} \tag{1}$$

$$\theta = \arctan\left(\frac{y}{x}\right) \tag{2}$$

with the polar axis (r) being the distance to the center of the image, and the polar angle (θ) being the angle of inclination. Next, the image needs to be resized to new dimensions. The width of the transformed image is equal to the amount of leds used in the led-strip, while the height is the angular resolution that we chose. Lastly the polar image needs to be encoded into the format and order that the LEDs require. Subsequently, the microchip on the display only needs to send the received data to the LEDs. This allows us to send data faster, and display images with a higher resolution.

3.2.2 Communication

We chose to do the communication with the display over a Wi-Fi connection, so it can be done using all kinds of devices without the need of a cable to upload images. To do this, we set up a Wi-Fi network on the microcontroller of the display, where clients can connect to. Multiple clients can connect to the access point at the same time. Since a picture is too big to send in a single HTTP POST request, we decided to set up a websocket, so there is an established connection between the device and the display, without much overhead. The display starts to host a websocket, to which clients can connect, as long as they are connected to the access point.

3.2.3 Application

To make the use of our display more user friendly, we decided to create an Android application. In the application, you can encode and send any image saved on your phone to the display with a single button. To create this, we had to port our Python code to Kotlin, and create an intuitive user interface.

Figure 2. KU Leuven logo displayed on the POV display.

4. Results

The encoding of multiple types of images works well, but when sending the encoded images to the display using the android application, sometimes a line would not arrive. The consistency of this happening depended on the different phones being used. There were no problems with the hardware, it functioned like expected. The final version of our POV display shows a high quality image that is stable enough to be easily recognizable, as shown in figure 2, but there is a slight fluctuation in its orientation.

5. Conclusion

Our final POV display works well, but there is room for improvement. The missing lines when sending images through the app is probably a hardware limitation, since using different phones with the same application gives different results. This could be solved by adding acknowledgements to negate data loss. The fluctuation of the orientation is a result from a bug in the timing of the display. We also made some parts of the casing a bit too high, which results in dark spots on the display if you look at it from an angle. If the code would get debugged and finetuned, it is possible to get even better results.

References

[1] Lewin Day. The basics of persistence of vision projects, 2019. Accessed: 26-5-2023.

[2] Seemit Praharaj. Led pov display. Accessed: 26-5-2023.

[3] ThomassVDD. Huge pov display, 2017. Accessed: 26-5-2023.