Deliverable J:

User Manual

Submitted by
B04C
Martin Matov, 300071043
Kristofer Grabovskis, 300065788
Emmanuel Nkwo, 8322442
Ryan Goodwin, 300079666
James Rhydderch, 300091810

Date 2019-12-4
University of Ottawa

Table of Contents

Introduction

Presentations are are a fundamental part of the office environment and the ability to fully understand its contents is crucial to those involved. Most presentations are created without integrating any of the accessibility options available in the software; this creates a problem for people with visual impairments as they will be unable to see the contents of the presentation. Furthermore, the accessibility settings on the most popular software are built to help as broad a spectrum as possible which means that they end up only being useful to a midmost part of the visual impairment spectrum.

Statistics Canada reports that 6% of Canadians have some form of visual impairment and it starts to really become a hindrance at the age of 43. According to Shared Services Canada, this indicates that across the nation there are thousands of office workers for whom slideshow presentations are a poor experience and these people end up having to take extra time out of their days to catch up on the presentation visuals.

Visi is a product that aims to solve this problem by bringing the presentation to a comfortable distance for the user. Visi is a device that takes the presentation materials (i.e the slideshow) and displays it on a screen in front of the user's eye. The device can be attached to a pair of glasses in order to allow the user to see the slideshow and still keep their focus on the presenter. The contrast of the display screen can be increased by switching from a transparent screen to an opaque one.

Current solutions to this problem rely on:

A. Increasing contrast and this only aids the user in seeing text - not images - which still cheapens the experience.
B. Relying on other senses, for example: screen readers which rely on the user's hearing but can still only provide a passable description of any photos or images, and once again do nothing to immerse the user in the presentations

Put simply, they are not given the same experience as their sighted colleagues. Also, the extra work needed for the addition of accessibility features - which should be standard practice - can cause some sighted presenters to harbor resentment for their colleagues with visual impairments. Finally, Visi is the better choice in terms of price as existing hardware solutions are much more expensive.

Important Features

Visi is a wearable head-ups display device that attaches to the users glasses that displays text in the users peripheral vision that the user cannot see clearly in a live presentation. The prototype is comprised of 2 electronic components, an Arduino Nano and a 0.96 inch OLED display. The choice for the Arduino Nano was because of its compact form factor while still providing sufficient power to handle multiple external components in the future. The microcontroller is connected to a 0.96 inch 128x64 pixel OLED display which was chosen because of the high contrast ratio it has and the small form factor. The display projects an image that is reflected by a mirror at a 45 degree angle from the display to a piece of clear acrylic where the image is visible in the users peripheral vision.

How it's Made

Housing

This is the external part of the device which holds all of the remaining components. The housing was 3D printed and thus is made of plastic. This method of production was chosen because it is well suited for the case which has well defined lines and few curves and also because it is inexpensive and rapid. The two halves are held together by printed attachment points.

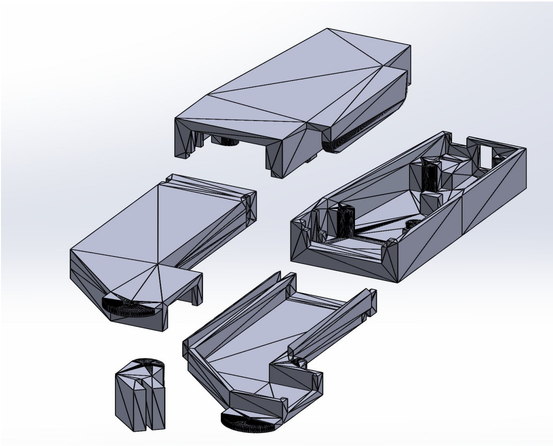


Figure 2: 3D printed housing of device

Internal Circuit

The internal circuit is made of and Arduino Nano, an OLED, a battery, and a bluetooth module.

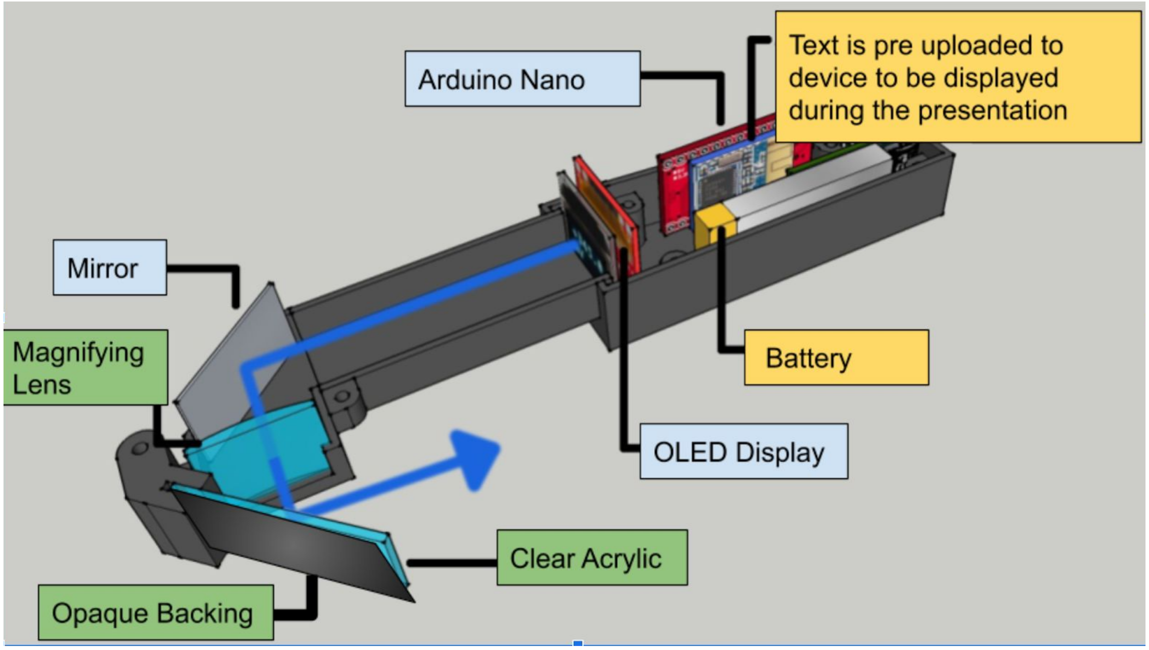


Figure 1: Internal components of Visi

Troubleshooting

Device not powering ON

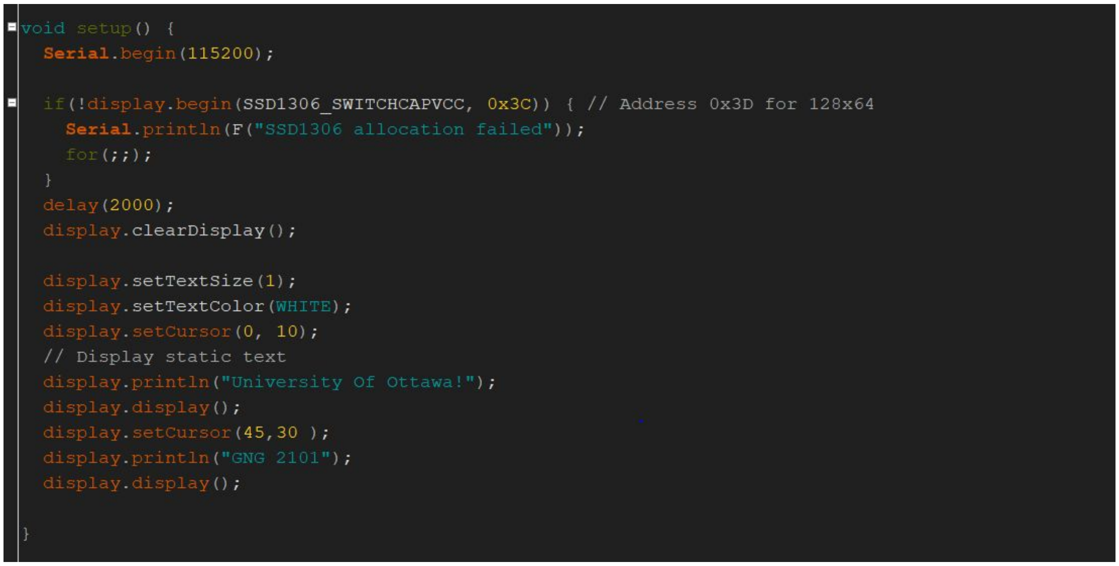
This most likely means that the battery has been depleted. You will need to replace your battery.

Display screen showing stars instead of my text

This is normal for these displays. The stars are simply there to give the screen time to initialize. This should happen for only about 15 seconds.

No text visible on the display screen

1. Carefully open the device to display the internal components
2. Connect the Arduino to a computer using the USB cord.
3. Download the Arduino IDE onto the computer from https://www.arduino.cc/
4. Go to https://makerepo.com/Emmanuel/gng-2101-group-b4c-accessible-presentations to download the text display code.
5. Look for the section shown in the image below

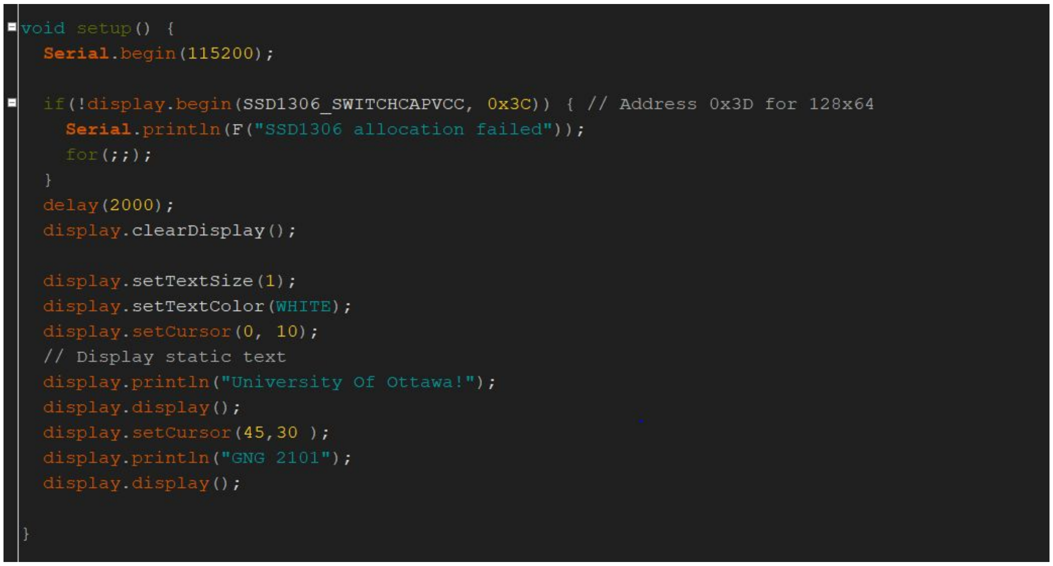


```
void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("University Of Ottawa!");
  display.display();
  display.setCursor(45,30 );
  display.println("GNG 2101");
  display.display();

}
```

6. Make sure that your text has been placed in the part of the code that says “display.println(“University of Ottawa”) as seen in the image below; replace only University of Ottawa - maintain everything else including the quotation marks.
7. Select ‘Upload’

I need to change the text color

1. Carefully open the device to display the internal components
2. Connect the Arduino to a computer using the USB cord.
3. Download the Arduino IDE onto the computer from https://www.arduino.cc/
4. Go to https://makerepo.com/Emmanuel/gng-2101-group-b4c-accessible-presentations to download the text display code.
5. Look for the section shown in the image below



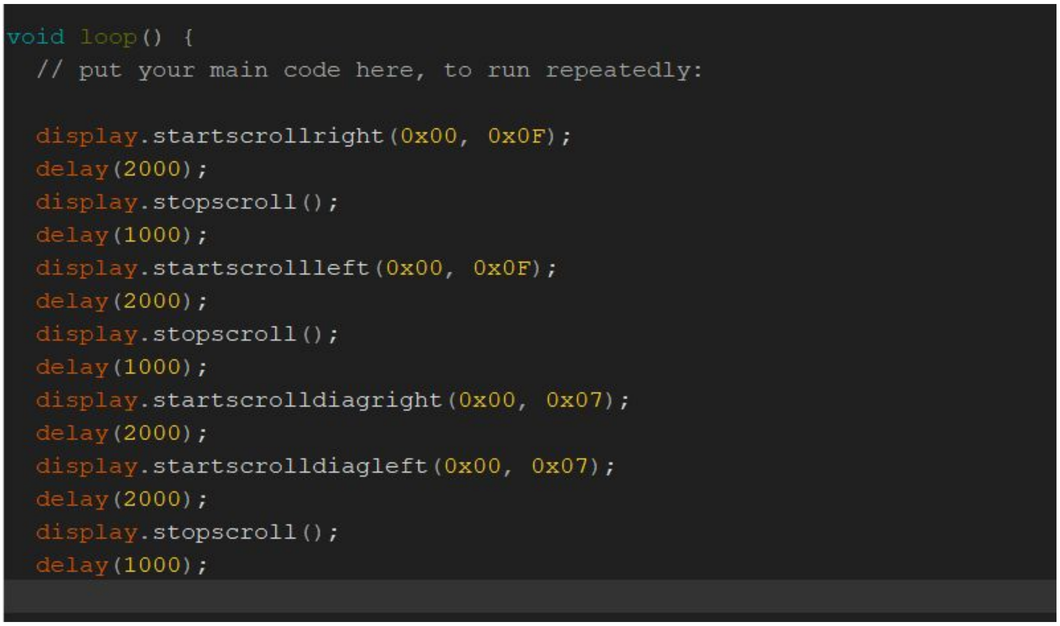
```
void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("University Of Ottawa!");
  display.display();
  display.setCursor(45,30 );
  display.println("GNG 2101");
  display.display();

}
```

6. Edit the text in brackets in the line that says “display.setTextColor(WHITE)”
7. Your new colour must be: RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, VIOLET.
8. Make sure the text is all in uppercase letters
9. Select ‘Upload’

Text is moving too slow

The display speed can currently only be changed by editing the Arduino code directly.

1. Carefully open the device to display the internal components
2. Connect the Arduino to a computer using the USB cord.
3. Download the Arduino IDE onto the computer from https://www.arduino.cc/
4. Go to https://makerepo.com/Emmanuel/gng-2101-group-b4c-accessible-presentations to download the text display code.
5. Look for the section shown in the image below



```
void loop() {
  // put your main code here, to run repeatedly:

  display.startscrollright(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
```

6. Edit numbers in the brackets in the lines that say “delay(2000)” or “delay(1000)”
8. Remember to maintain the 2:1 ratio between “delay(2000)” and “delay(1000)”
9. Select ‘Upload’

I edited the code and now the device is unresponsive

In the event that the device becomes unresponsive after you edit the code, the default code can be found here: https://makerepo.com/Emmanuel/gng-2101-group-b4c-accessible-presentations ;

1. Carefully open the device to display the internal components
2. Connect the Arduino to a computer using the USB cord.
3. Download the Arduino IDE onto the computer from https://www.arduino.cc/
4. Go to https://makerepo.com/Emmanuel/gng-2101-group-b4c-accessible-presentations to download the text display code.
5. Open the file and select “Upload”.

Future Recommendations

We recommend that the housing for the device be improved upon, the current design fails to accommodate the Arduino Nano in a manner that makes the USB port accessible and as such the casing must be opened in order for the user to connect to the arduino and edit the code. In the same vein, a better, more compact battery should be used. The current battery is a simple 9V Alkaline battery. This means that it cannot be placed inside the case otherwise the design for the case will be to large, bulky or otherwise unwieldy. We recommend a small rechargeable battery be used possibly a lithium ion battery; this way the need for the user to open the case will be further reduced - when the battery is depleted, they can simply plug it in instead of removing it entirely.

In terms of functionality, we recommend that the magnification be made customizable. This can be achieved by making the lens to translate linearly between two set points. This will allow more people to make use of the device while reducing the need for individual customization. Further, the display screen should be attached in such a way that it is easy to remove. This means that users can be given both the clear screen and the opaque one and can use either as they see fit without the need for any tools or worse, a second device. The last functionality recommendation is the OLED screen. This will need to be greatly improved in order for the device to display entire presentations and meet the goal of equalizing the experience between visually impaired and sighted people. A screen with a much higher resolution will be needed - 1280 x 720 at least - to properly display presentation materials and maintain all the transitions, effects, and images.

Finally, we recommend that an app be built to accompany the device. With the app the user should be able to upload the files to be displayed. It should also enable them to edit the formatting of the display; they should be able to change the position, speed, size, and colour of the information being displayed. With this, the user will no longer need to edit the Arduino code directly and thereby reducing the possibility of the user causing errors within the code.

Conclusions

There are many takeaways from this project and it has enabled us to learn very much. First, client feedback is crucial. The client most likely understands the need far better than anybody and as such they must help in guiding the design of the various prototypes. That said, we also learned that the client may not understand what is possible within the constraints and it is our job to explain it to them. We also learned that there is often not as much time as you think. The use of workflow diagrams and scheduling tools allowed us to stay on track for most of the deliverables but when tasks were improperly or unrealistically scheduled work broke down. There was a sharp drop in effectiveness when tasks were started later than they were supposed to or when unplanned events interfered.

Moreover, we learned the importance of rapid -and frequent - prototyping. The use of 3D printed materials in our prototypes definitely saved our group time and money. It also allows frequent changes of even the smallest details which in turn allows you to create a superior final product. The inexpensive microcontroller was also a bonus; it allowed us to experiment with various mounting and connecting techniques (during which we damaged two of the three boards we purchased). These rapid prototyping and testing solutions allow the team and the product to move forward much faster and are certainly strategies to be employed on any future projects.

All of these techniques and strategies that we learned over the course of building our product allowed us to build a functional product. They were crucial in allowing us to finish the various deliverables on schedule and will definitely be employed in any engineering projects we undertake going forward.