FabEYE

by Sabrina Aboulkacem

AR-Glasses with Arduino and external Device OR a Bluetooth-Controller for your Arduino

Overview

FabEYE are open-source augmented reality glasses which send their state to an external device and receive commands from it.

FabEYE-Terminal is an application which uses Qt and is tested on OS X. It would also be possible to run it on Linux (with BlueZ 4.x/5.x), Android and iOS, which are the supported platforms for Bluetooth in Qt [1], but it would require a redesign of the user interface for mobile devices.

Required Materials

Electronics

- Arduino (I used an Arduino MEGA 2560 because it has more RAM)
- USB-Cable (to upload the code to your Arduino)
- Bluetooth 2.0 –Module (I am using an HC-06)
- 6 Push-Buttons
- 2 I2C-displays which use SSD1306 (make sure you can use two different I2C-addresses)
- a lot of wires and a breadboard

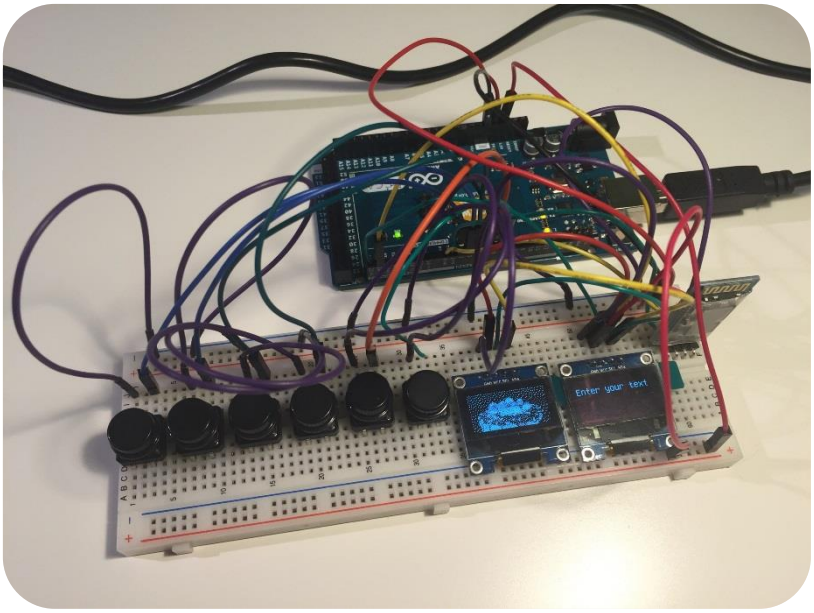
Software:

- Arduino IDE [2]
- Qt Open Source [3]
    - If you want to compile it for use on your iPhone or Mac, you have to use Qt on your Mac and install Xcode as well.

Libraries:

- U8glib [4]

Wiring the Arduino

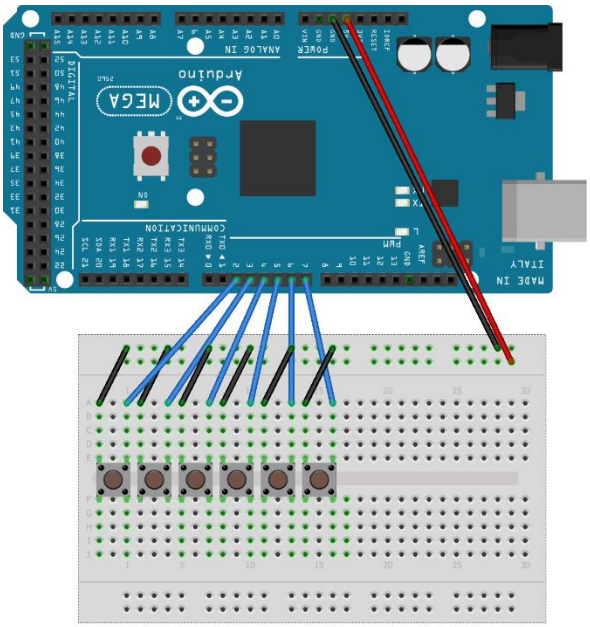


Wiring the Bluetooth-Module

- Connect the RX-pin of the Bluetooth-module to the TX-pin of the Arduino(pin 1)
- Connect the TX-pin of the Bluetooth-module to the RX-pin of the Arduino(pin 0)
- Connect VCC/+5V of the Bluetooth-module to 5V of the Arduino
- Connect GND of the Bluetooth-module to GND of the Arduino

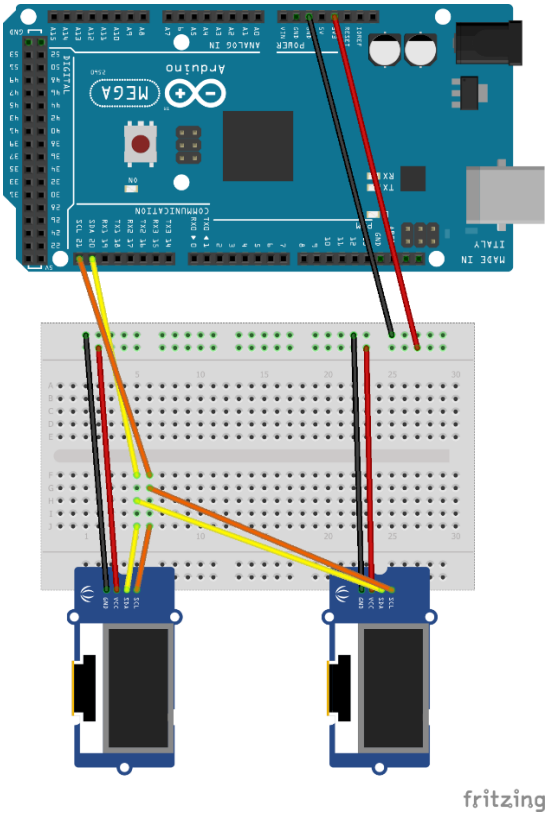
Wiring the Buttons

Connect the buttons to pin 2, 3, 4, 5, 6 and 7 and ground. (You can change the pins in the `button_init` array in “ar_glasses.h”)



Wiring the Displays

- Connect VCC of the displays to 3.3V of the Arduino
- Connect GND of the displays to GND of the Arduino
- Connect SCL of the displays to SCL of the Arduino
- Connect SDA of the displays to SDA of the Arduino



Setting up the Arduino

Installing the IDE:

Download the Arduino IDE [2] and install it on your pc.

Installing the Library:

Download the U8glib [4] (https://github.com/olikraus/U8glib_Arduino) library and install it:

1. Download the U8glib from: https://github.com/olikraus/U8glib_Arduino
2. Remove „-master“ from the name of the downloaded zip-file
3. Open your IDE and navigate to the menu bar -> Sketch -> Include Library -> Add .ZIP Library
4. Select the downloaded zip-file and import it to your IDE

If the installation fails for any reason, unzip the library and copy it to
[YourUserName]/Documents/Arduino/libraries

Adjust the Library:

U8glib uses the I2C address 0x78. We need to change that, so we can use two or more different displays.

1. Open “u8g_com_arduino_ssd_i2c.c”
2. Change
   ```
   #define I2C_SLA          (0x3c*2)
   ```
   to
   ```
   uint8_t I2C_SLA = 0x3c*2;
   ```

Setting up FabEYE

1. Open up the FabEYE-project in the Arduino IDE
2. Change the I2C-addresses in the `display_init` array in “ar_glasses.h” (“`OLED_LEFT_I2C`”, “`OLED_RIGHT_I2C`”) to your displays’ addresses.
3. If you do not have six buttons or two displays, adjust `button_init` and/or `display_init` arrays in “ar_glasses.ino”.

Upload FabEYE

1. Open the FabEYE-project in the Arduino IDE
2. Make sure the Bluetooth-module is not connected, to prevent interferences between the Bluetooth-module and the transmission via USB
3. Connect your Arduino to your PC
4. Navigate to menu bar -> Tools and make sure that the right board, processor and port are chosen
5. Upload it to your Arduino

Setting up the external device

If your PC uses OS X, you can use the compiled “FabEYE_App.app” file. Otherwise you can recompile it for other platforms:

1. Download and install Qt Open Source [2]
2. Build the “ar_glasses” library

3. Open FabEYE-Terminal and update `LIBS` (path of the compiled library) and `INCLUDEPATH` (path of the header file of the library) in “FabEYE_Terminal.pro”.
4. Build FabEYE-Terminal and press Run

FabEYE (Arduino)

Class AR_Component:

Every part of FabEYE is an instance of AR_Component (including the FabEYE itself). An AR_Component instance has an id and provides the methods `send_status`, `send` and `receive`.

- `send_status`
  - sends all constant values (e.g. a display's dimensions or the number of buttons)
- `send`
  - sends messages (e.g. button is pressed, captured picture from camera)
- `receive`
  - handles the received messages/commands (e.g. picture or text to be drawn on a specific display)

This architecture makes it possible to expand FabEYE with more components and also to choose which components are used.

List of components:

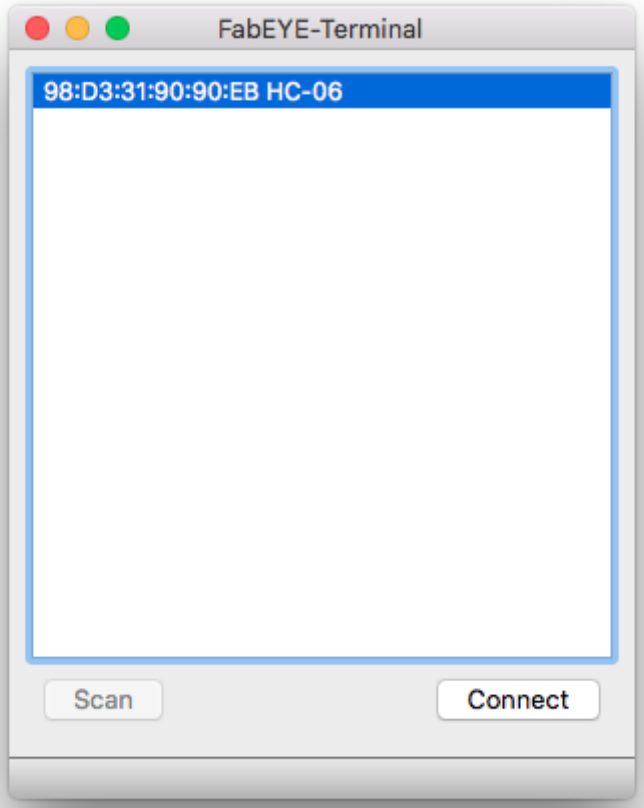
- FabEYE (id=0):
  - Stores component groups (It is possible to choose which components and how many of these exist)
  - `send_status` goes through all component groups and sends their type (1=Button, 2=Camera, 3=Display), number of contained components and status.
  - `send` goes through all component groups, sends their type and calls their send method
  - `receive` checks the receiver of any incoming message
    - If the message is sent to a component, the function will pass it to the component group that contains the receiver
    - FabEYE accepts commands to
      - send the state
      - stop or start sending messages
- Class **AR_Buttons** (id=1):
  - `send_status` and `receive` do nothing since buttons have no constant properties and cannot receive any commands
  - The `send` method sends info if the button is pressed or not
- Class **AR_Cameras** (id=2):
  - `send_state`: Width and height of the pictures that the camera can capture
  - Not yet implemented
- Class **AR_Displays** (id=3):
  - `send_state`: Width and height of the display
  - The `receive` method can receive
    - a text and draw it on the screen
    - a bitmap-picture (full screen) and draw it on screen
    - a bitmap-picture and a position and draw it at this position (not yet implemented)
- Class **AR_Components** (Component groups):
  - Stores instances of AR_Component
  - Can send the number of components stored in it
  - Any call to `send_status` and `send` goes through all the stored components and calls their respective methods.
  - The `receive` method calls the receiver's `receive` method

FabEYE-Terminal Application

The FabEYE-Terminal is an example application that communicates with FabEYE. Its task is to look for Bluetooth devices, retrieve their information and send texts or pictures to a selected device.

1. Connecting to a device

Press the “Scan”-Button to scan for devices and select a device from the list. The list shows the devices’ Bluetooth-addresses and names. Press “Connect” to connect to the selected device.



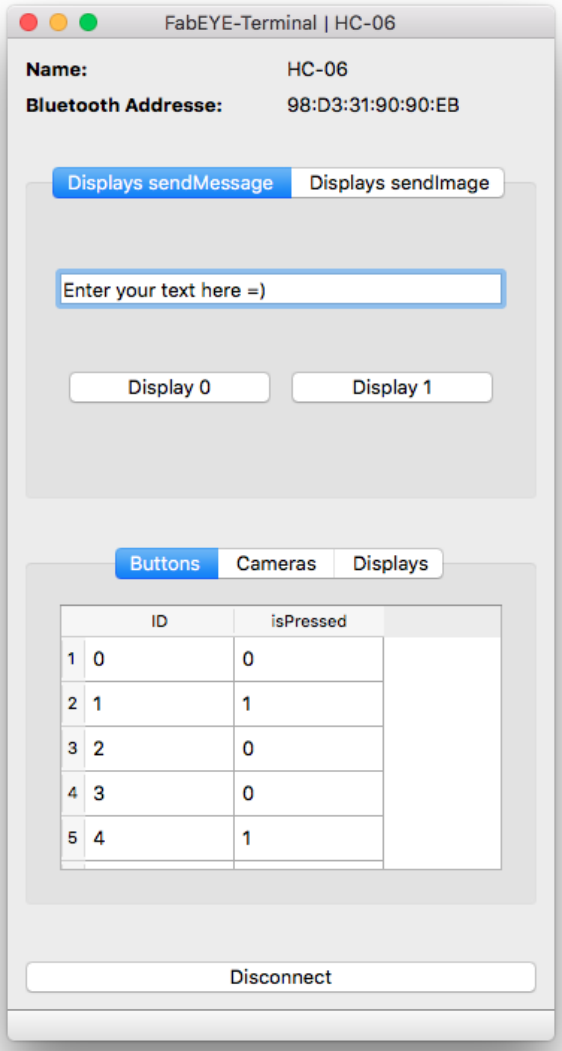
2. The Terminal

Connection info

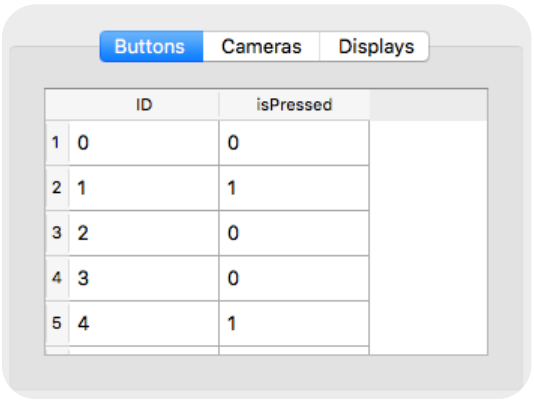
Send texts or images

Get status of components

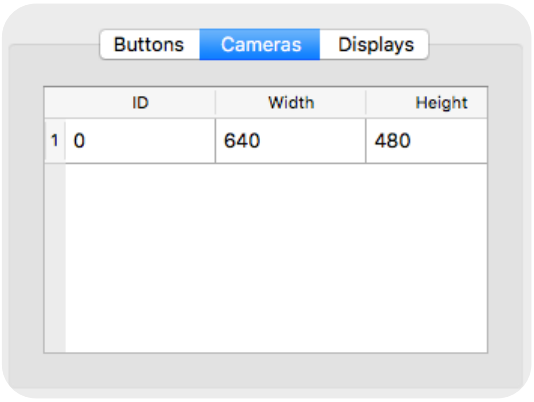
Disconnect from device



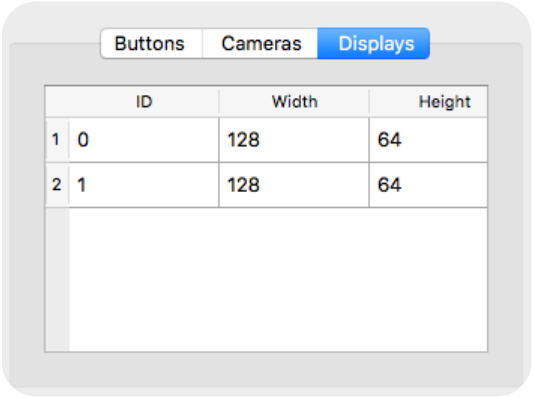
Component-status



Button-status: shows if a button is pressed or not (updates real-time)

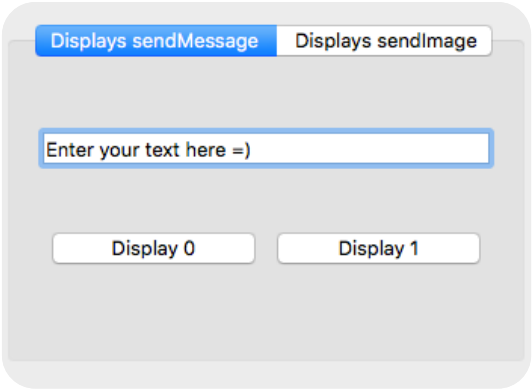


Camera-status: shows the widths and heights of the pictures the cameras can capture

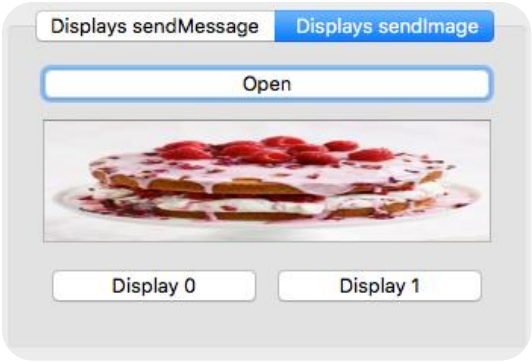


Display-status: shows the widths and heights of the displays

Sending messages



Enter a text in the textbox and press a button. A text will be sent to and drawn on the specified display.



Press “Open” and choose an image to be sent. It will then be previewed in the box below. Press one of the buttons to send the image. It will then be drawn on the specified display.

How to create a new component?

How to create a new component for the Arduino-Application?

1. Use the template “component_template.cpp” and “component_template.h” for creating a new component. Use the comments to change it to your needs.
2. Define a code for your component in “ar_protocol.h”
3. Add a case in `init_component()` in “ar_glasses.ino”

How to create a new component for the Qt-Application?

1. Use the template “component_template.cpp” and “component_template.h” for creating a new component.
2. Add your new component to “ar_components.h”
3. Add your new component to `initComponent()` in “ar_glasses.cpp”

For sending a message to the FabEYE, use the signal

```
void sendMessage(const uint8_t &cmd_type, const QByteArray &message)
```

If an attribute has changed, use the signal

```
void statusChanged(const uint8_t &status)
```

How to change the used components or adjust their attributes?

You only need to change the used components in your Arduino-Application.

1. Open the “ar_glasses.ino”-file
2. Adjust the `button_init`, `camera_init`, `display_init` arrays and/or add required arrays.
3. Adjust the `ar_components` array.
4. Adjust `init_ar_components()`.

Casing (Concept)

Materials

- 3mm plywood
- 1mm transparent Plexiglas for prisms and “button breadboard”
- Faux leather and foam for cushion
- 4cm elastic band to hold the glasses on the head
- Velcro tape to hold the cushion on the front of the glasses
- Screws and female screws
- 1 hinge
- 1 toggle switch
- 1 DC power plug
- 1 battery clip
- Glue for wood and Plexiglas

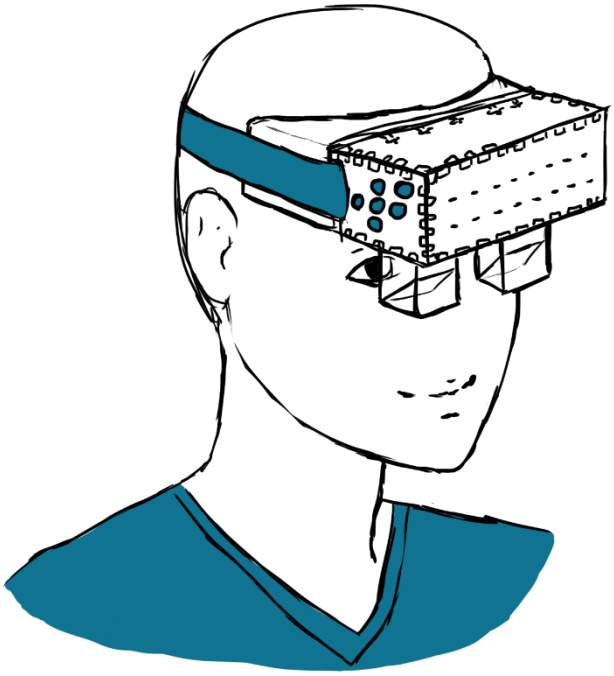
Concept

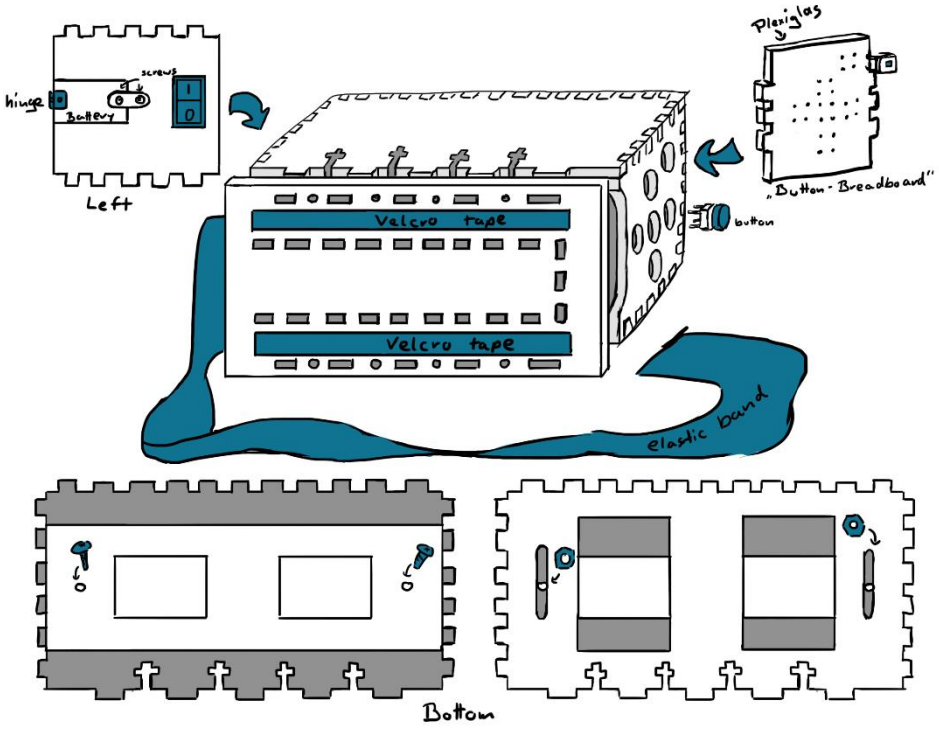
The FabEYE casing consists of an inner and an outer part.

Outer part

The outer part is basically a cuboid with one removable side (attached with T-slots) to access the inner part. There are holes for the buttons on the right side and a power switch and a hinged lid for the battery on the left side. On the bottom side there are two holes for the prisms.

For wearing comfort there is a cushion attached to the removable side using Velcro tape. The cushion is made from faux leather and foam.



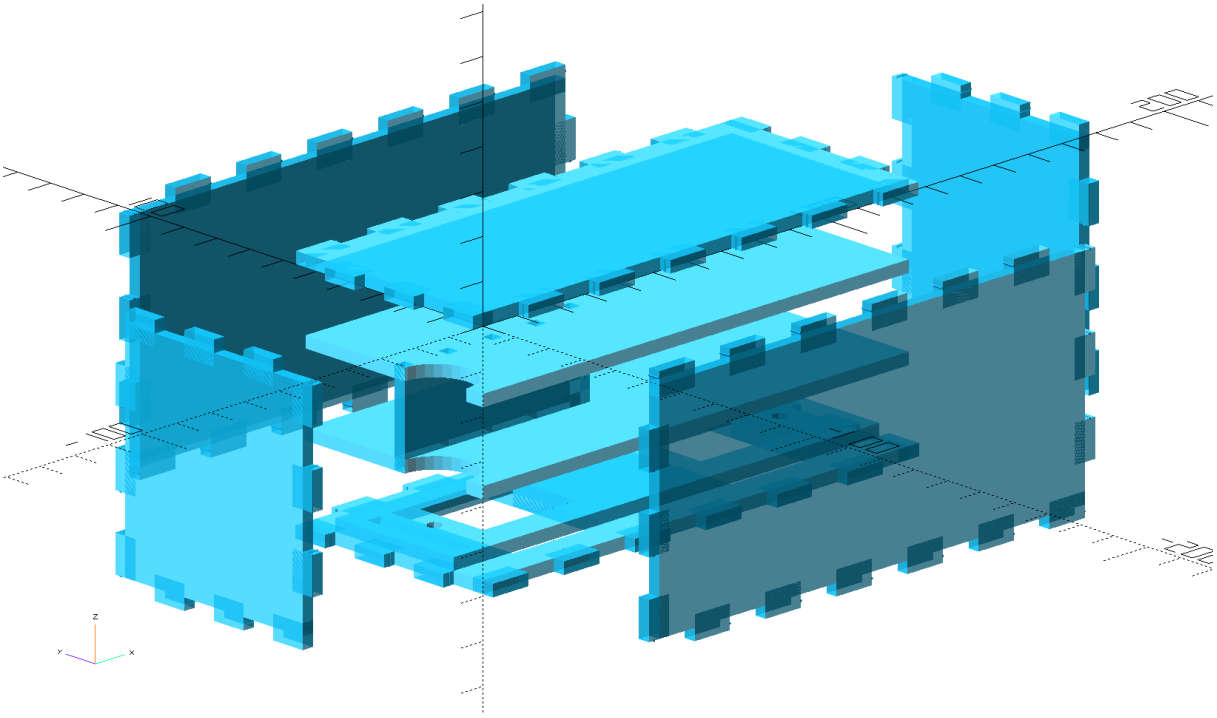


Inner part

The inner part is made of three platforms and a custom breadboard to hold the buttons. The bottom platform holds the displays and prisms and is attached to the bottom side of the outer part using two screws and can be positioned closer to or further away from the eyes.

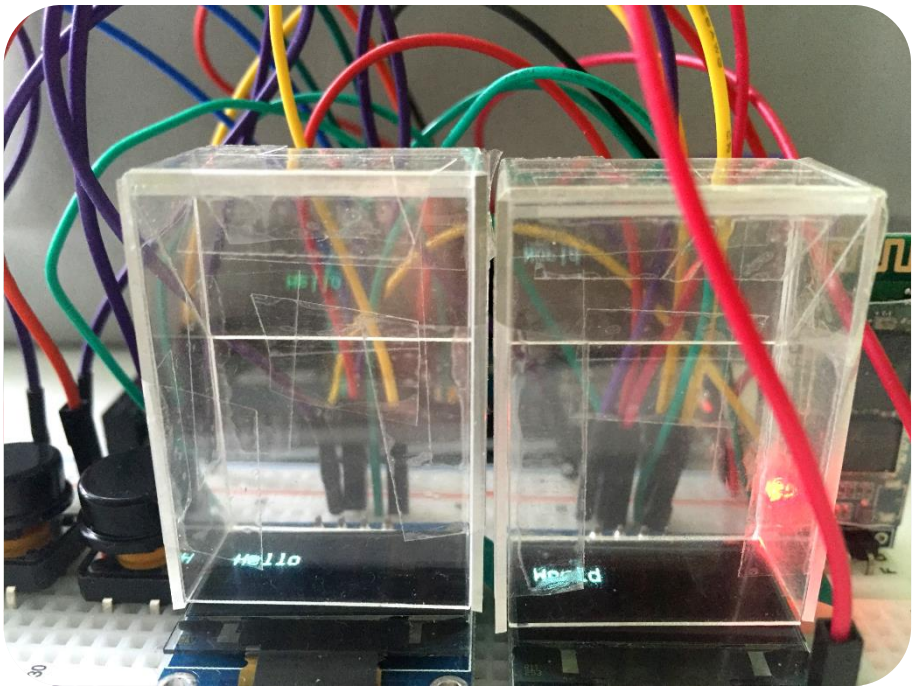
The middle and top platforms and the breadboard are attached to the front and back side using fingers to hold them in place. The Arduino is placed on the top platform and the middle one holds the battery and the Bluetooth module.

The breadboard is cut from Plexiglas and holds the buttons. It is placed on the right side.



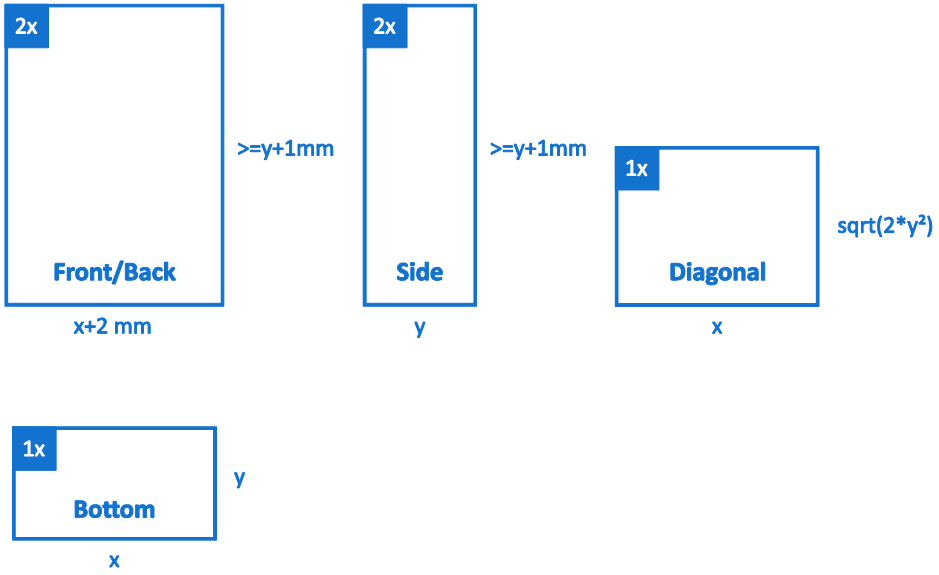
Prisms

The prisms are cut from 1mm transparent Plexiglas and can be assembled with glue. They are placed below the displays and are used to reflect the light from the display into the person's eyes.



Cosntruct your own Prisms

You can easily construct your own prisms with your own measurement. If you have a display with width=x and height=y you can use this guidance to cut your own prism:



Known Bugs

- If you reconnect to FabEYE with the application, the program could crash. Reset your Arduino and restart the app, if that happens. It looks like the states are not sent correctly or the App does not wait till nothing is sent anymore, so it is not initialised correctly.
- The app crashes if the connection is unexpectedly aborted because it is not trying to reconnect.
- You can still use the “Connect”-Button, when nothing is selected

What is next?

- Fixing reconnect and connect bugs
- Creating a casing
- Adding the functionality to switch on mirroring on a specified display
- Adding the functionality to create textboxes
- Using Bluetooth 4.0 to make it possible to use an iPhone
- Creating an alternative user interface for mobile applications

References

[1] Qt Bluetooth documentation: http://doc.qt.io/qt-5/qtbluetooth-index.htm
[2] Arduino IDE: https://www.arduino.cc/en/Main/Software
[3] Qt Open Source: https://www.qt.io/download-open-source/
[4] U8glib: https://github.com/olikraus/U8glib_Arduino