Touch-Free Museum Quiz Exhibit

Practical (2)

Author: Joseph Manfredi Cameron

Abstract

This project's aim was to create a museum exhibit where visitors can participate in and interact with a multiple-choice quiz without having to touch anything. Visitors can achieve this solely by showing their hands to a camera to answer questions and by hovering their hands over a photoresistor sensor to perform confirmation actions. The main motivation for creating this exhibit to be touch-free and to not require any contact with buttons or the like was to develop a museum exhibit that can reduce the potential for contamination spread. The need for this type of exhibit in busy public settings such as museums has been highlighted by the COVID-19 global pandemic where it is necessary to limit the spread of the virus. This exhibit was developed using the p5.js JavaScript software library [1] and the Arduino Uno microcontroller [2]. The software developed in p5.js acts as both an input and an output for the exhibit, where a screen displays the quiz screen as an output, and a camera collects images from which the software can detect the number of fingers a visitor is holding up. Similarly, the hardware controlled by the Arduino also serve as inputs and outputs of the exhibit.

Interaction Overview

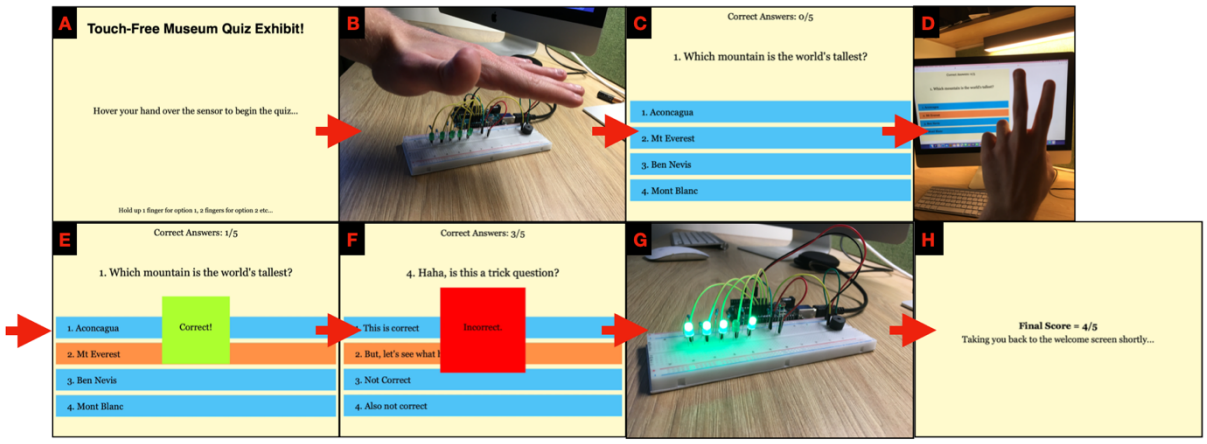

Figure 1

Figure 1 shows the standard flow of interaction with this touch-free exhibit. Figure 1A shows the initial welcome screen visitors are faced with when first approaching the exhibit. As detailed on the screen, visitors simply hover their hand over the sensor (which is a photoresistor) to start the quiz. This is shown in Figure 1B. Following a short countdown, the user is then presented with a question as shown in Figure

1C. To select an option, the user holds up the corresponding number of fingers as the option number. For example, selecting option 1 would require the user to hold up one finger. Figure 1D shows a user selecting option 2 by holding up two fingers. To lock in the selection and proceed to find out if the answer was correct or not, the user simply hovers their hand over the photoresistor again as shown in Figure 1B. Following this action, the user is presented with a feedback dialog box that reveals if the selection was correct or not. If it was correct, the user is presented with the dialog box shown in Figure 1E, along with a pleasant beep sound from the Piezo speaker connected to the Arduino microcontroller. If the selection was incorrect, the user is instead presented with the dialog box shown in Figure 1F, along with an erroneous-like sound from the Piezo speaker connected to the Arduino microcontroller. For each question a user gets correct, the corresponding green LED that's connected to the Arduino microcontroller lights up. Figure 1G shows an example of a user who has got questions 1, 2, 3, and 5 correct, but question 4 incorrect. The user then repeats this described process for every question in the quiz. Once visitors finish their quiz, they are presented with the screen shown in Figure 1H which shows their final score. Then after a few seconds, the exhibit returns to the welcome screen shown in Figure 1A and waits for another visitor to start the quiz.

Parts List

Hardware

- Arduino Uno Microcontroller x1
- Arduino Uno USB Power Cable x1
- Breadboard x1
- Photoresistor x1
- Passive/Piezo Buzzer x1
- Green LED x5
- 10K Ohm Resistor x1
- 220 Ohm Resistor x5
- Breadboard Cables x11
- Display Screen x1
- Camera x1
- 10 x 10K resistors
- 10 universal remote controls (to hack for IR signals)
- (If you want to enable alternative and traditional interaction, you can add a mouse and keyboard or a touchscreen because my software also allows for it)

Other

- Arduino IDE
- P5.js
- Ml5 Handpose Machine Learning Model
- P5.serialcontrol (from handling serial connection from Arduino to P5)

Process

The first thing to do to recreate this exhibit is to create the following circuit (shown in various forms by Figures 2 and 3) on a breadboard and connect the hardware components to pins on the Arduino microcontroller.

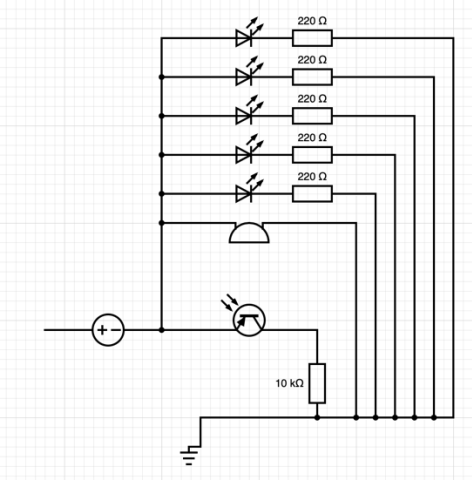

Figure 2: Circuit Diagram

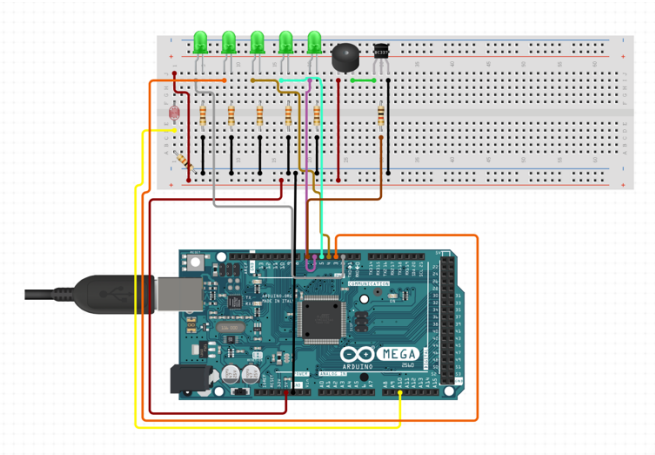

Figure 3: Example Wiring Diagram

Note that the wiring diagram shown in Figure 3 illustrates the basic connectivity of the components together on the breadboard and the Arduino. The exact circuit I wired up and used is shown in Figure 4.

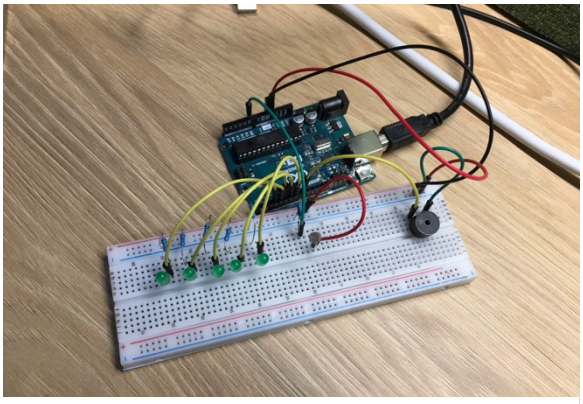

Figure 4: My Exact Circuit Setup

The hardware connected to the Arduino is used for both input to the exhibit and p5.js, and output from the exhibit and p5.js. I used a photoresistor sensor to act as an input sensor where visitors can confirm selections by hovering their hands. The act of hovering a hand over the sensor results in less light reaching the photoresistor, which produces a different signal. A photoresistor was used over other sensors such as microphones etc. because I deemed it to be the most appropriate for the setting of a potentially busy and noisy museum environment. For example, if a microphone was used to detect phrases from visitors, the presence of any background noise (likely in a museum) may interfere with the detection and lead to a frustrating interactive experience. A piezo buzzer is used as an output speaker to provide immediate feedback to users on the correctness of responses, and 5 green LEDs are also used to act as outputs that provide sustained feedback for which questions were correct or incorrect.

With this circuit set up, the code found in Appendix A is then uploaded to the Arduino Uno microcontroller. As evident in the code, all the capability for Arduino serial communication to and from p5.js is set up here. Getting to this point represents a milestone with respect to hardware setup for this project.

The next step is to develop the quiz software in p5.js. The use of a multiple-choice style quiz became prominent, as the main goal of this project is to interact with a quiz without touching objects and the idea of holding up fingers to correspond with either option 1, 2, 3, or 4, became an intuitive way of selecting the desired option on the quiz. Appendix B shows the JavaScript code that contains my method of counting how many fingers a user holds up via camera images. My method of counting involves identifying the fingers and then comparing the distances between each of their start and end points.

Hence, this method can be quite robust in a busy museum setting as it does not rely on a set distance between the user and the camera, it will still count fingers regardless of how far away users hold up their hands. The software determines where the fingers are by using the ml5 Handpose machine learning library and model [3]. Figure 5 shows an illustration of what my method plus the ml5 Handpose model detects through the camera. There are limitations in terms of how accurately the model can identify hands and fingers, but if users stand within a reasonable range of distance from the camera, the model performs surprisingly well.

Figure 5: Illustration of points found on hand.

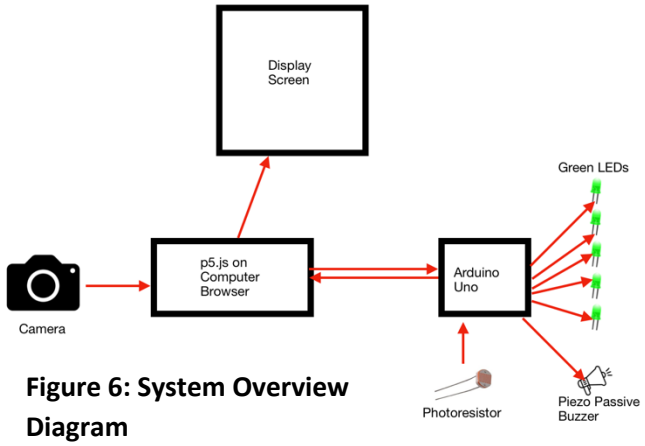

Figure 6: System Overview Diagram

Being able to count the number of fingers a user holds up represents another milestone reached in this project. From this point, it's simply a case of building a desirable quiz in p5.js that can also write messages over the serial port to the Arduino microcontroller that can trigger the hardware components that were setup earlier. Appendix C shows an example of the code that sends serial messages to the Arduino for triggering both the piezo buzzer sounds, and the corresponding LEDs to turn on for correctly answered questions. Note how these serial messages correspond to the serial messages that the Arduino expects to receive in the code shown in Appendix A.

Now after completing all these steps, the exhibit should have a similar system overview as the system overview displayed in Figure 6.

Challenges, Key Points & Conclusion

Overall, this museum quiz exhibit does indeed allow users to interact with a quiz while not touching any physical objects, which would satisfy certain requirements demanded by the current global pandemic. Furthermore, both the Arduino and p5.js have been successfully utilised to act as both inputs and outputs of this exhibit.

One of the biggest challenges of this project was gauging the correct sensitivity that all the input sensors should have in order to provide a pleasant interactive experience to users. At the start of the process, it

was very apparent that the ml5 Handpose model out of the box with no modifications didn't have any intuitive way to count fingers. Instead it simply assigned points to the image where it predicts the fingers are. Therefore, I had to develop an algorithm that counts the number of fingers held up based on the difference in finger lengths that the model finds. Also, the model tended to be too reactive which proved frustrating to use, so I had to smooth that out for less jitter in the input.

On a similar note, much experimentation was required to get the correct threshold for the photo resistor at which point it detected the right amount of light that a hovering hand would allow.

An obvious avenue for future work if more time allowed would involve creating a better machine learning model, or technique, for counting the fingers a person is holding. The current method still requires a very precise gesture from the user and will occasionally get the reading wrong if a user is casually holding up fingers rather than exaggerating the gesture as required.

Works Cited

[1] L. L. McCarthy, Q. Ye, E. Masso and Open Source Community, "p5.js," Processing Foundation, [Online]. Available: https://p5js.org. [Accessed 21 March 2022].

[2] "Arduino Microcontrollers," Arduino, [Online]. Available: https://www.arduino.cc. [Accessed 21 March 2022].

[3] "ml5 Handpose," ml5.js, [Online]. Available: https://learn.ml5js.org/#/reference/handpose. [Accessed 21 March 2022].

Appendix A

P2-ArduinoCode

```
// Arduino Code for Hands-Free Museum Quiz Exhibit
// Author = 200036885

int sensorValue;

// For reading serial messages from p5
int incomingByte;

// Green LED for Q1
const int greenLED1Pin = 3;
// Green LED for Q2
const int greenLED2Pin = 4;
// Green LED for Q3
const int greenLED3Pin = 5;
// Green LED for Q4
const int greenLED4Pin = 6;
// Green LED for Q5
const int greenLED5Pin = 7;

void setup() {

  Serial.begin(9600);

  pinMode(greenLED1Pin, OUTPUT);
  pinMode(greenLED2Pin, OUTPUT);
  pinMode(greenLED3Pin, OUTPUT);
  pinMode(greenLED4Pin, OUTPUT);
  pinMode(greenLED5Pin, OUTPUT);


  digitalWrite(greenLED1Pin, LOW);
  digitalWrite(greenLED2Pin, LOW);
  digitalWrite(greenLED3Pin, LOW);
  digitalWrite(greenLED4Pin, LOW);
  digitalWrite(greenLED5Pin, LOW);

}

void loop() {

  // Sending light transistor data to P5
  sensorValue = analogRead(A0);
  int mappedValue = map(sensorValue, 1023, 0, 0, 255);
  Serial.write(mappedValue);

  // Check for data coming from P5
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    if (incomingByte == 'a') {
      correctTone();
    } else if (incomingByte == 'b') {
      incorrectTone();
    } else if (incomingByte == 0) {
      // Turn on LED 1
      digitalWrite(greenLED1Pin, HIGH);
    } else if (incomingByte == 1) {
      // Turn on LED 2
```

```
      digitalWrite(greenLED2Pin, HIGH);
    } else if (incomingByte == 2) {
      // Turn on LED 3
      digitalWrite(greenLED3Pin, HIGH);
    } else if (incomingByte == 3) {
      // Turn on LED 4
      digitalWrite(greenLED4Pin, HIGH);
    } else if (incomingByte == 4) {
      // Turn on LED 5
      digitalWrite(greenLED5Pin, HIGH);
    } else if (incomingByte == 'c') {
      // Reset all LEDs to be off
      digitalWrite(greenLED1Pin, LOW);
      digitalWrite(greenLED2Pin, LOW);
      digitalWrite(greenLED3Pin, LOW);
      digitalWrite(greenLED4Pin, LOW);
      digitalWrite(greenLED5Pin, LOW);
    }
  }

}

// Plays a nice beep sound for correct answers
void correctTone() {
  tone(8, 2000, 20);
}


// Plays an error sound for incorrect answers
void incorrectTone() {
  tone(8, 1000, 20);
  delay(250);
  tone(8, 500, 20);
}
```

Appendix B

JS countFingers.js > ...

```
1   // countFingers.js
2   // Author = 200036885
3   // This file contains helper functions for
4   // counting the number of fingers held by users.
5
6   function getDigit(points) {
7       if (points.length > 0) {
8         const x1 = points[0][0];
9         const y1 = points[0][1];
10        const x2 = points[points.length-1][0];
11        const y2 = points[points.length-1][1];
12        return calculateFingerLength(x1, y1, x2, y2);
13      }
14    }
15
16    function calculateFingerLength(x1, y1, x2, y2) {
17      let x = x2 - x1;
18      let y = y2 - y1;
19      return Math.sqrt(x * x + y * y);
20    }
21
22    function countFingers(fLengths) {
23      let count = 0;
24      fLengths.pop(); // Gets rid of empty last reading
25      fLengths.shift(); // Remove thumb digit from reading (gets rid of first reading)
26      if (fLengths.length > 0) {
27        let longest = Math.max.apply(Math, fLengths);
28        fLengths = fLengths.map((fLength) => getPercentage(fLength, longest));
29        for (let i = 0; i < fLengths.length; i++) {
30          if (fLengths[i] > 60) {
31            count++;
32          }
33        }
34        return count;
35      } else {
36        return 0;
37      }
38    }
39
40    function getPercentage(x, y) {
41      return ((x/y) * 100);
42    }
43
```

Appendix C

```
311  // Check if answer is correct or not
312  function checkAnswer(tag) {
313    if (tag) {
314      currentResult = "yes";
315      visualFeedback = new response(tag, canvasWidth, canvasHeight);
316      currentScore++;
317      // Write to Arduino through serial
318      // Lets arduino play the correct tone
319      serial.write('a');
320      serial.write(currentQuestionIndex);
321    } else {
322      currentResult = "no";
323      visualFeedback = new response(tag, canvasWidth, canvasHeight);
324      // Write to Arduino through serial
325      // Lets arduino play the incorrect tone
326      serial.write('b');
327    }
328    setTimeout(() => { nextQuestion(); }, 1000); // Leave a second between questions
329  }
```

```
207  // Change state of quiz to the welcome screen
208  function showWelcomeScreen() {
209    serial.write('c');
210    countdown = false;
211    quizActive = false;
212    quizOver = false;
213    welcome = true;
214  }
```