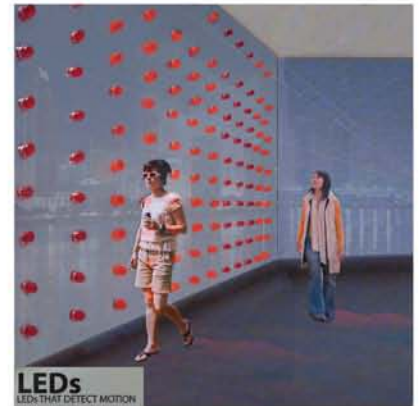


CHANGE-MY-VIEW wall module

Katherine Chin | Making Things Interact



This image shows how my model of LEDs could potentially be implemented. The LEDs turn off when a person walks in front of the wall.

PROJECT SUMMARY

This project explores dynamic boundaries between spaces and how walls can change the relationship between behavior and the environment. I proposed to create a wall that is opaque when no one is there, and becomes transparent as the person is walking along to create a window of a view. Because of the price of smart glass, I used LEDs to demonstrate this effect. This wall responds to human activity and requires human interference to function. By moving within the space, people can redefine the boundaries between public and private, inside and outside.

Each module of the wall becomes transparent when a person walks in front of it so that he/she can look through and enjoy the view!

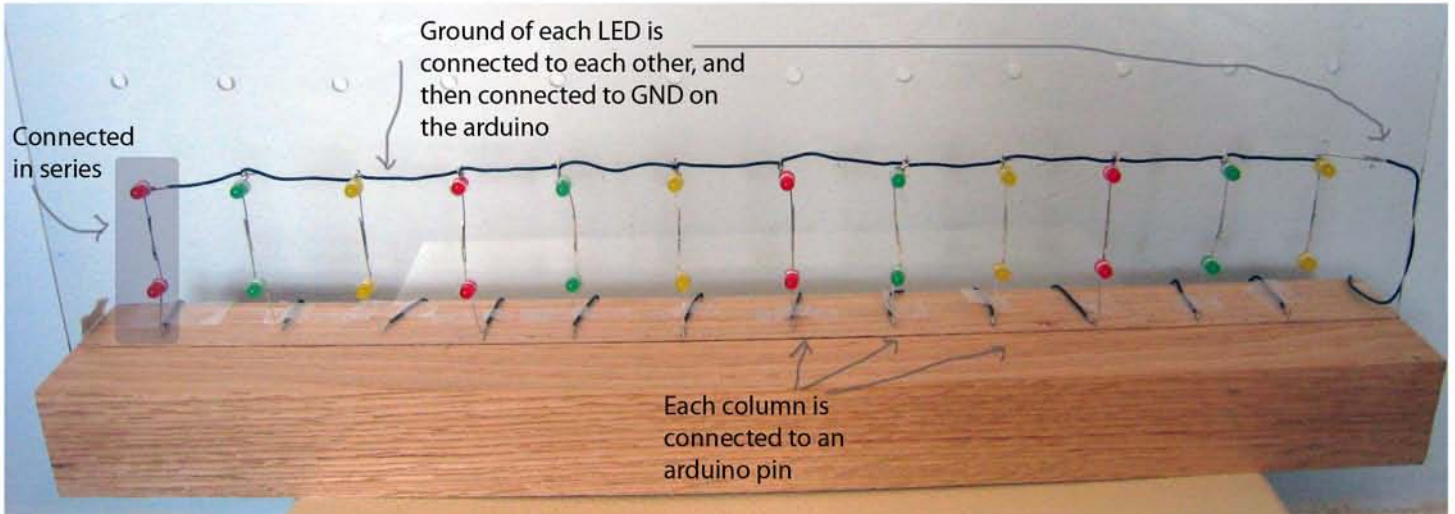


The glass switches from transparent to opaque by changing the current going through the glass.

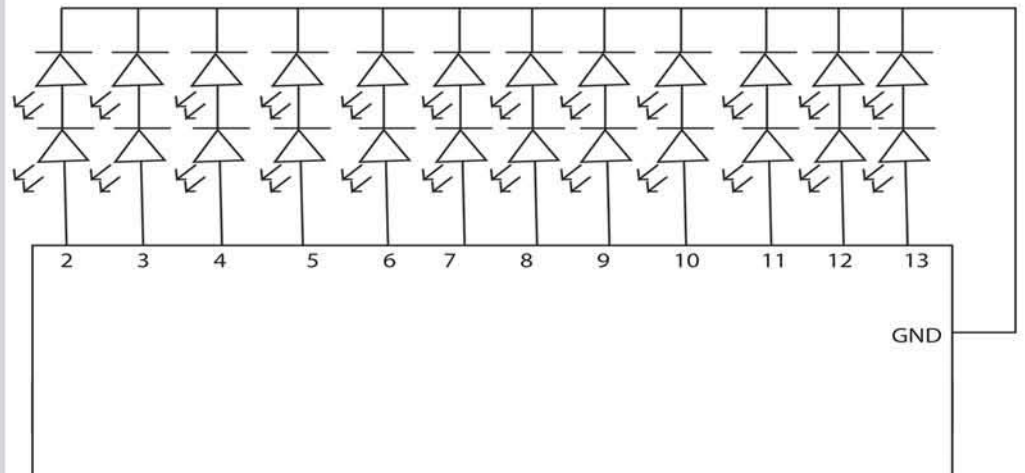
CHANGE-MY-VIEW wall module

Katherine Chin | Making Things Interact

Photograph of physical model with notes on connections:



Schematic design diagram of electronics



PARTS LIST:

Webcam
LEDs
Plexi Glass
Wood as stand
Solder
Wire
Arduino

Additional Information:

Ways to improve the project include:

- Using LEDs that can change color
- Using range finders to detect the distance people are from the wall
- Being more "intelligent", it could have memory and replay patterns
- Use an LED matrix

CHANGE-MY-VIEW wall module

Katherine Chin | Making Things Interact

Processing Code - Frame differencing

```
int movementSum = 0; // Amount of movement in the
frame
int numPixelsArray [] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

for (int i = 0; i < numPixels; i++) {
  color currColor = video.pixels[i];
  color prevColor = previousFrame[i];

  int currR = (currColor >> 16) & 0xFF;
  int currG = (currColor >> 8) & 0xFF;
  int currB = currColor & 0xFF;
  int prevR = (prevColor >> 16) & 0xFF;
  int prevG = (prevColor >> 8) & 0xFF;
  int prevB = prevColor & 0xFF;
  int diffR = abs(currR - prevR);
  int diffG = abs(currG - prevG);
  int diffB = abs(currB - prevB);
  int sum = diffR + diffG + diffB;
  movementSum += sum;

  for (int iter = 0; iter < 12; iter++)
  {
    if (i%640 < 53.333*(iter+1) && i%640 > 53.333*iter)
      numPixelsArray[iter] += sum;
  }
  .....

  for (int iter = 0; iter < 12; iter++)
  {
    if (i%640 < 53.333*(iter+1) && i%640 > 53.333*iter)
      numPixelsArray[iter] += sum;
  }

  if (movementSum > 5000000) {
    updatePixels();

    int largestI = 0;
    int largest = numPixelsArray[0];
    for (int i = 1; i < 12; i++)
    {
      if (numPixelsArray[i] > largest)
      {
        largest = numPixelsArray[i];
        largestI = i;
      }
    }
    port.write('A' + largestI);
  }
}
```

Arduino Code (for writing one pin 'A')

```
char val;
int LEDpinA = 13;

void setup(){
  pinMode(LEDpinA, OUTPUT);
}

void loop(){

  if (Serial.available()) {
    val = Serial.read();
  }
  if (val == 'A') {
    digitalWrite(LEDpinA, HIGH);
    digitalWrite(LEDpinB, LOW);
    digitalWrite(LEDpinC, LOW);
    digitalWrite(LEDpinD, LOW);
    digitalWrite(LEDpinE, LOW);
    digitalWrite(LEDpinF, LOW);
    digitalWrite(LEDpinG, LOW);
    digitalWrite(LEDpinH, LOW);
    digitalWrite(LEDpinI, LOW);
    digitalWrite(LEDpinJ, LOW);
    digitalWrite(LEDpinK, LOW);
    digitalWrite(LEDpinL, LOW);
  }
}
```