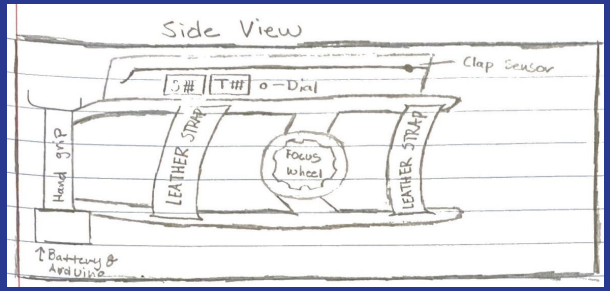
Wearable Focus Clapper

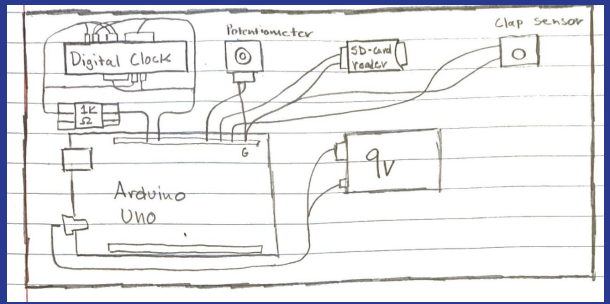
By: Brett Rabbiner, Roman Lynch, and Gage Gruidel

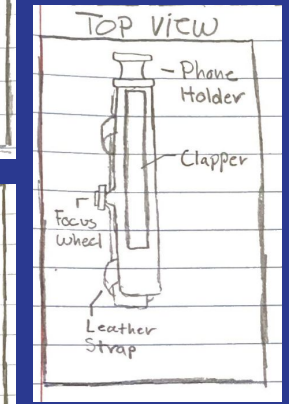
Video Demo

Phase 1: Design

- Most movies today still use clappers. But on indie production sets, its use can be a hassle, since there aren't dedicated focus pullers and production assistants. This device allows the focus puller to have the clapper on their person at all times, along with the focus wheel, seamlessly integrating the two jobs into one handy device that you can't misplace. This could help increase production efficiency, and standard use of the clapper on indie production sets.
- The specific function of the wearable clapper is to allow the user to have the ability to focus and use the clapper at the same time on movie sets with ease.







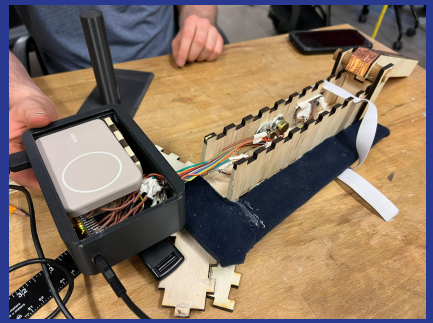
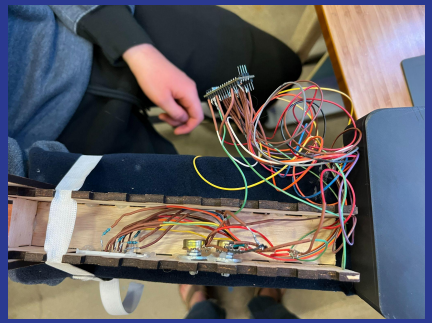
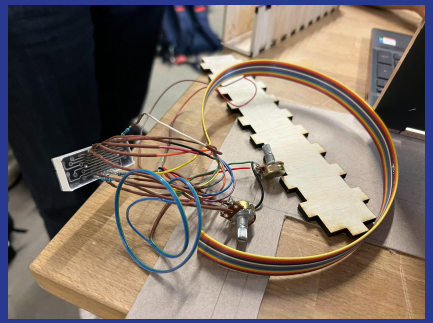
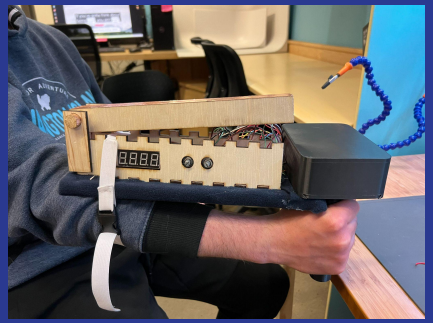
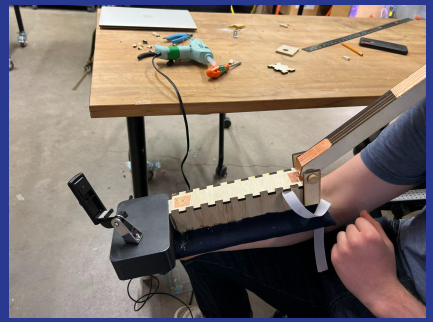
Phase 2: Prototype

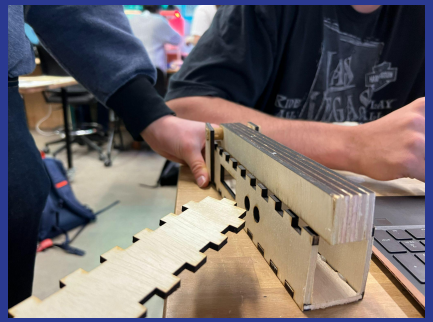
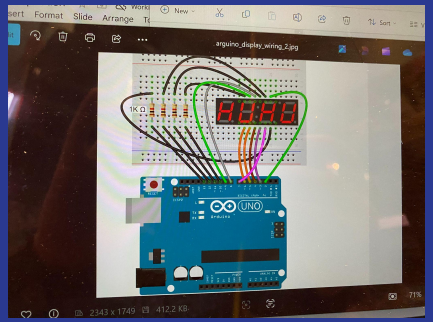
- When prototyping we started with a cardboard design and some plywood to make the clapper. We then made the base shape of our design with insertions for our wiring, potentiometers, scene/take display, and a button to trigger the device in the clapper. We also added a strap and a makeshift handle.
- The first prototype was successful but we needed to find a better way to trigger the scene change, add housing for a power source, a better handle, and add comfort for your arm.
- We took our first prototype into consideration when designing our final build but made a lot of improvements to the design.

Phase 3: Build

- The final build we laser cut the clapper with pre cut holes in the file for a cleaner design. We added a butt hinge we made out of laser cut pieces for the “clap.” We also added a foam piece the length of a forearm for comfort, an improved 3D printed handle and battery holder along with a built in stand for your phone.
- Our build required the use of laser cutting the clapper, sewing the strap, 3D modeling the battery holder/handle, finalizing our working circuit, and lastly adding our phone holder.
- We filtered the data in our code mainly in the use of the potentiometers just making sure the output with the knobs used would be able to scroll 1-99 and display the correct values for the clapper display.
- To meet the power requirements for our circuit and add ease for the user we decided to use a rechargeable power bank as our power supply that has 5 volts with 2.4 amps, output to multiple sources. This allows rechargeability and gives the necessary output for our device.
- We did not get to our stretch goal of functionality with an SD card but our team was happy with how the final product turned out.

Documentation: Photos/Diagrams

Documentation: Code

```
#include "SevSeg.h"
SevSeg sevseg;

// ANLOGUE PINS
const int potLastTwo  = A0;
const int potFirstTwo = A1;
//const int saveButton  = A2;

// START VALS
int sceneNum = 1;
int takeNum  = 1;

// QA?QC (debounce) VARS
int buttonState = 1;
int lastButtonState = 1;
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;
```

```
void setup() {
  // SETUP 7-SEG LED CLOCK
  byte numDigits= 4;
  byte digitPins[] = {10, 11, 12, 13};
  byte segmentPins[] = {9, 2, 3, 5, 6, 8, 7, 4};

  sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins, true);
  sevseg.setBrightness(90);

  //pinMode(saveButton, INPUT_PULLUP);

  Serial.begin(9600);
  Serial.println("Setup complete");
}
```

```
void loop() {
  // READ POTENIOMETERS
  int potVal1 = analogRead(potFirstTwo);
  int potVal2 = analogRead(potLastTwo);

  // MAP POTENIOMETERS from (0-1023) to (1-99)
  sceneNum = map(potVal1, 0, 1023, 1, 99);
  takeNum  = map(potVal2, 0, 1023, 1, 99);

  // DISPLAY SCENE NUM and TAKE NUM to clock (must be one number)
  int displayVal = sceneNum * 100 + takeNum;
  sevseg.setNumber(displayVal, 0);
  sevseg.refreshDisplay();

  /* CODE FOR SAVE BUTTON (not implemented)
  int reading = digitalRead(saveButton);
  if ((millis() - lastDebounceTime) > debounceDelay) {
    // If the button state changed and is now pressed (falling edge)
    if (lastButtonState == 1 && reading == 0) {
      Serial.println("TIME CODE SAVED");
    }
  }
  // Save the current state as the last state for next loop
  lastButtonState = reading;
  */
}
```

Thank you!