00Bruin

Design Review Presentation

Edwin Lin, Tyler Newton, Hasti Salsali, Alex Lopez

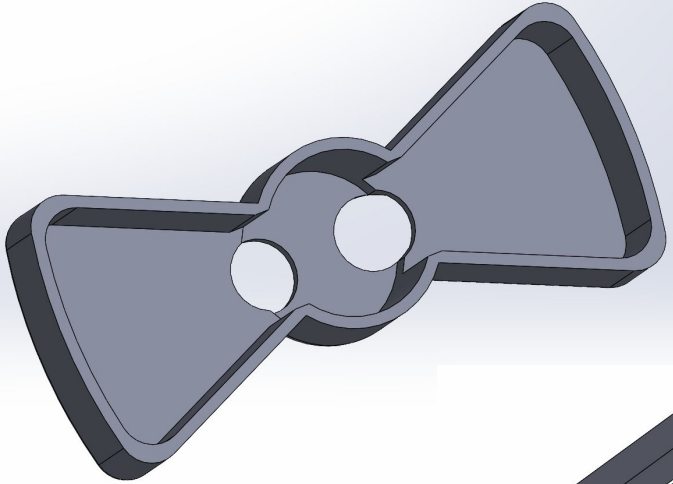
Our Approach

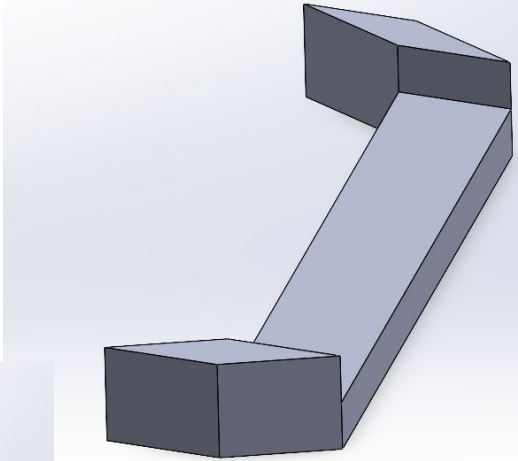

00Bruin
HAcK
Alex Lopez
Edwin Lin
Hasti Salsali
Tyler Newton

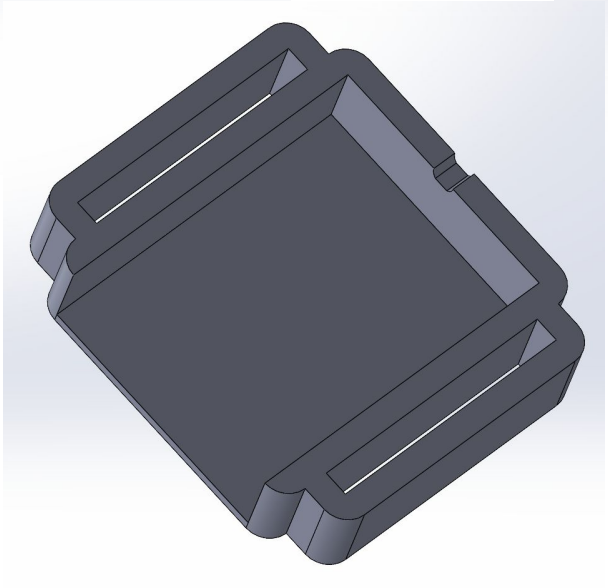

Temp: 75.7 F
Humidity: 47%
Light: 0
Hello HAcK

Mechanical Components - Tyler

- **CAD Designs:** All designed in SOLIDWORKS
- **3D Printing:** 3 pieces were 3D printed
  - Bowtie body: to house the components in the bowtie
  - Bowtie hooks: to allow the bowtie to be strapped on and worn
  - Watch body: to house the components of the watch
- **Laser Cutting:** The faceplate of the watch was laser cut out of wood to hold the device together and mark it with our super cool logo 😎
- **Other Fabrication:** Power tools, fabric, velcro



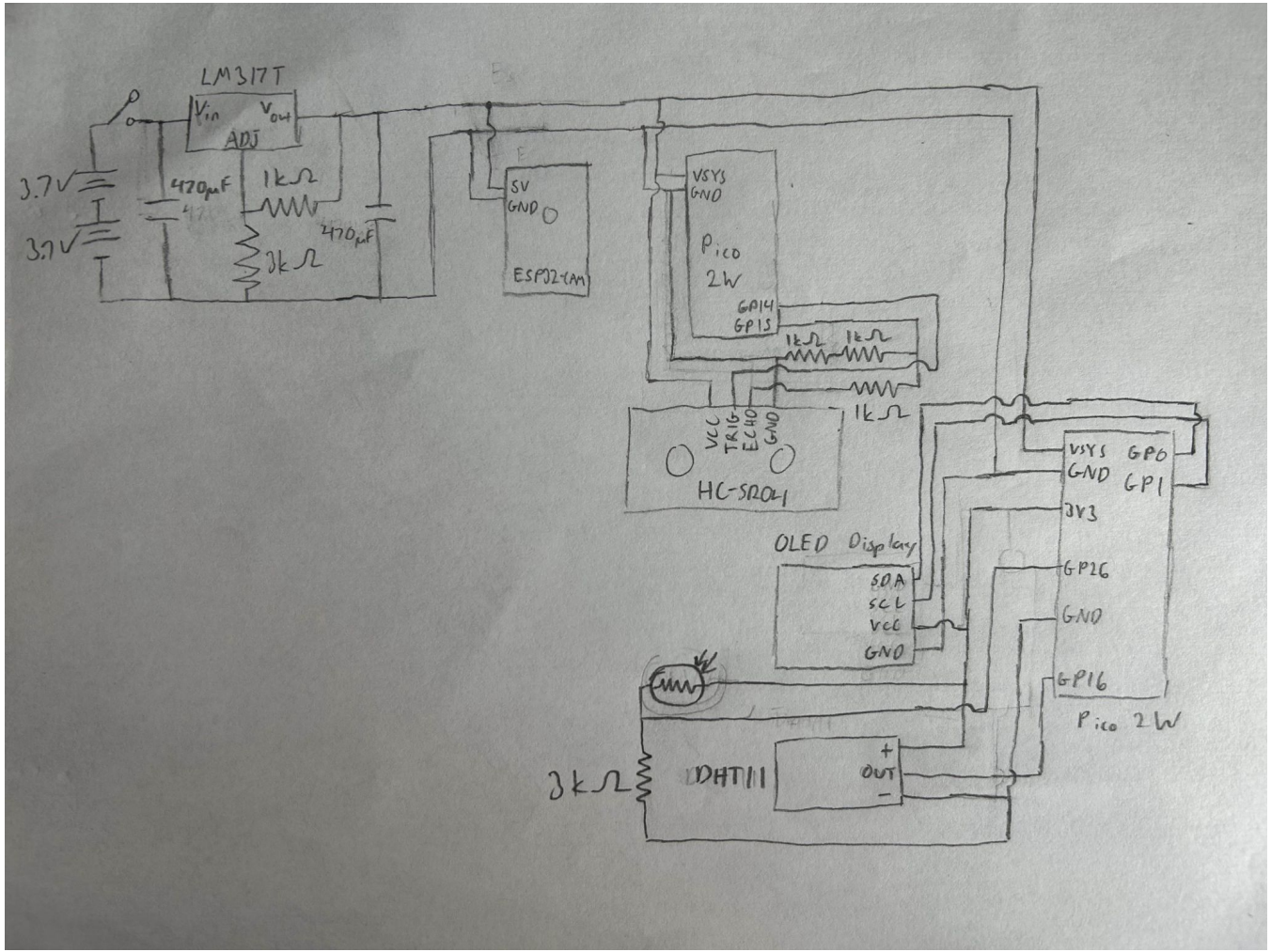




Electronics and Schematics - Edwin

- **Raspberry Pi Pico 2W:** handles sensors used and send information to OLED Display and to backend
- **ESP32-CAM:** takes photos to send to operator
- **Batteries and Voltage Regulator:** combined batteries and voltage regulator to reduce voltage from 7.4 V to a consistent 5.1 V

Circuit Diagram



Website Code - Hasti

- **Version Control:** Github was used to save, share and maintain history of code.
- **Website Development:** The website was built using JavaScript.
- **Frontend-Backend Connection:** Socket.io communication protocol was used to establish communication between the frontend and backend.

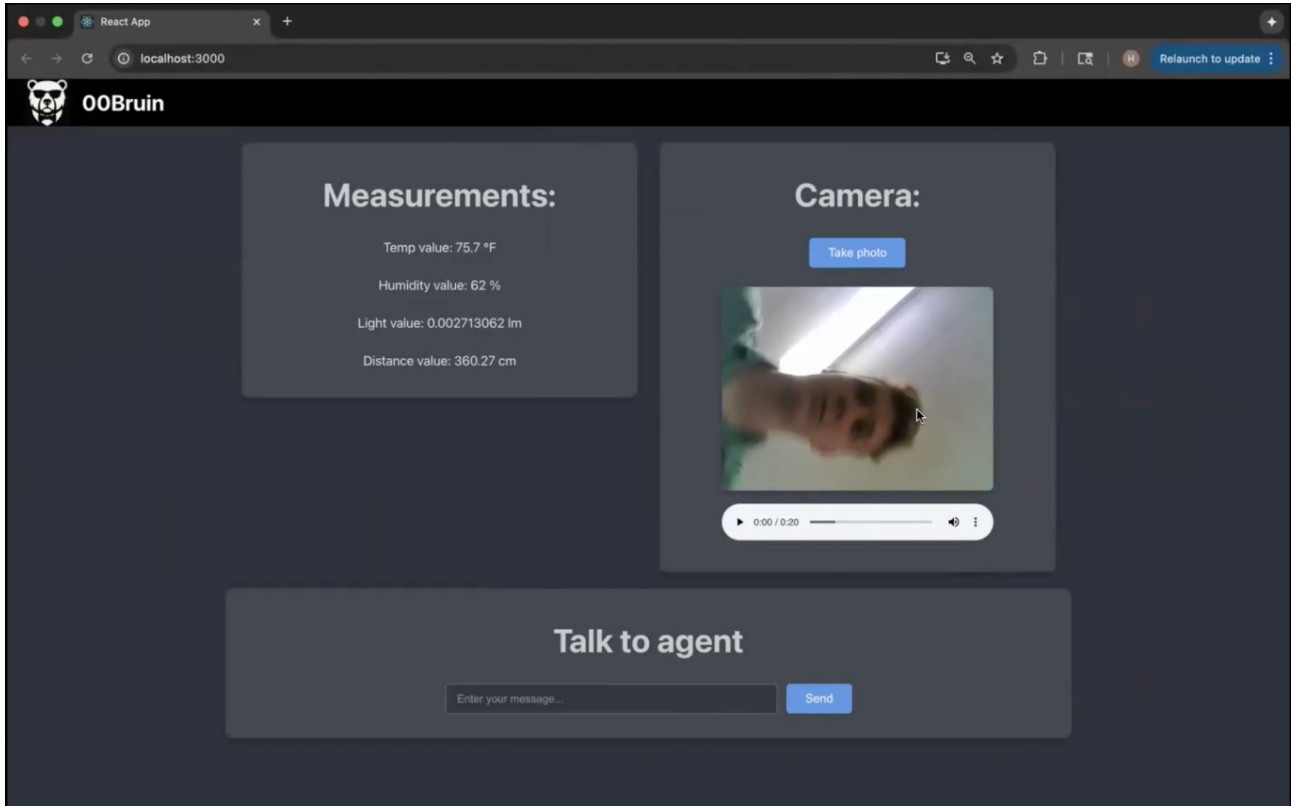
```
//UPDATING MEASUREMENTS:

const [tempRes, settempRes] = useState("No measurements done yet!");
const [humidRes, sethumidRes] = useState("No measurements done yet!");
const [lightRes, setlightRes] = useState("No measurements done yet!");
const [distRes, setdistRes] = useState("No measurements done yet!");

  //geting values from backend:
socket.on('temp', (latestTemp) => {
  settempRes(latestTemp);
  //console.log("got temp measurements" + latestTemp)
})
socket.on('humidity', (latestHumidity) => {
  sethumidRes(latestHumidity);
  //console.log("got Humidity measurements" + latestHumidity);
})
socket.on('light', (latestLight) => {
  setlightRes(latestLight);
  //console.log("got light measurements" + latestLight);
})
socket.on('ultrasonic', (latestUltrasonic) => {
  setdistRes(latestUltrasonic);
  //console.log("got distance measurements" + latestUltrasonic);
})
```

AI API - Hasti

- **AI API Calls:** Python was used for all OpenAI API interactions.
- **Image Analysis Workflow:**
  - An initial API call was made to analyze and describe photos.
  - A subsequent, separate API call converted the resulting text description into audio.

```python
#AI PHOTO ANALYSIS FOR PICTURE:
response = client.responses.create(
    model="gpt-4.1-mini",
    input=[{
        "role": "user",
        "content": [
            {"type": "input_text", "text": "Describe photo"},
            {
                "type": "input_image",
                "file_id": file_id,
            },
        ],
    }],
)

print(response.output_text)


#AI TEXT TO SPEECH:
speech_file_path = os.path.join(script_dir, "../frontend/public/image_to_speach.mp3")
instruction = """Accent/Affect: heavy posh british accent; """

with client.audio.speech.with_streaming_response.create(
    model="gpt-4o-mini-tts",
    voice="ash",
    input= response.output_text,
    instructions= instruction
) as response:
    response.stream_to_file(speech_file_path)
```

Website Demo



Watch’s Raspberry Pi Pico Code - Alex

- 3 main scripts were used in conjunction with the Spy watch we developed.
  - dht.py: This script creates a base class that lets a microcontroller talk to the DHT11 by sending and receiving electrical signals to read and check the sensor data.
    - Downloaded from GitHub Repo: https://tinyurl.com/y2crnymm
  - ssd1306.py: This code is a library that allows a microcontroller to control a small OLED screen, letting it display text, shapes, or graphics using I2C or SPI connections.
    - Downloaded from GitHub Repo: https://tinyurl.com/4bjwdk5j

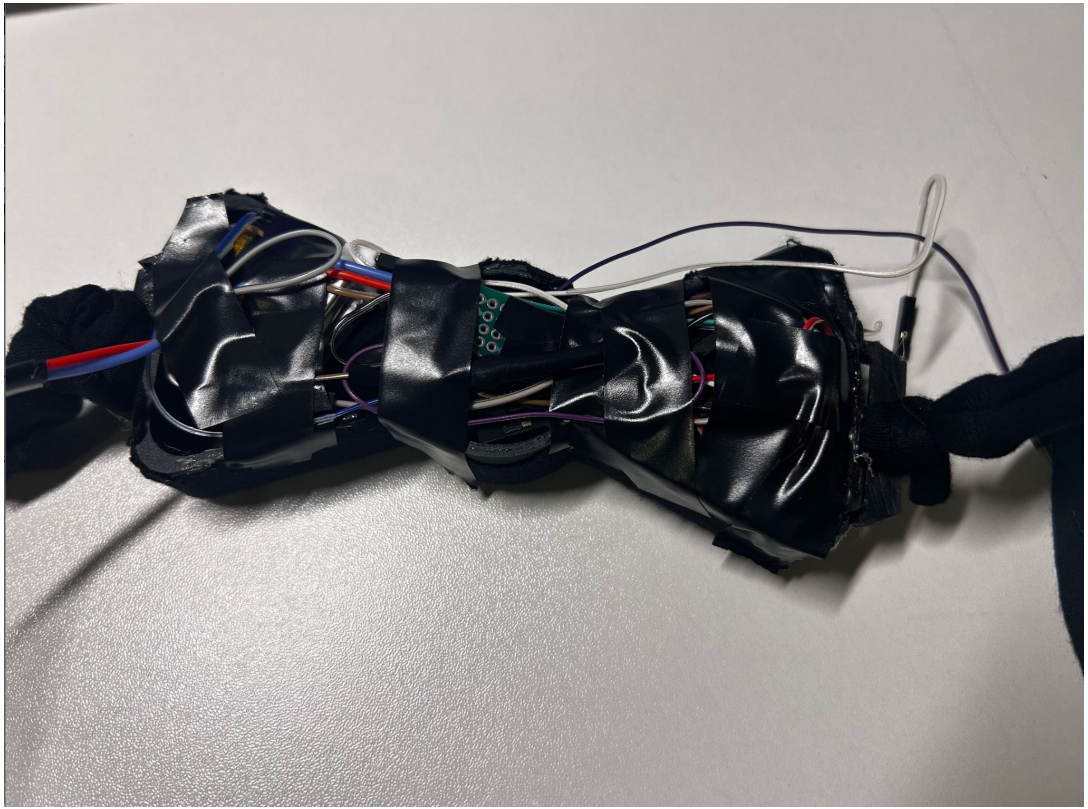
Watch’s Raspberry Pi Pico Code - Alex

- main.py: This script mainly collects data from the DHT11 temp/humidity sensor and Photoresistor within the watch device and displays those values on the OLED.

```
while True:
    oled.fill(0) #Clear OLED display

    if show_clock:
        #Show current time
        oled.text("Time (PST):", 0, 10)
        oled.text(format_time(), 0, 30)
        oled.show()
        time.sleep(1)

    else:
        try:
            #Measure temperature and humidity from DHT11
            sensor.measure()
            temp_c = sensor.temperature

            #Convert Celsius to Fahrenheit, adjust by -2 (calibration offset)
            temp_f = temp_c * 9 / 5 + 32 - 8.5
            hum = sensor.humidity

            #Read calibrated light sensor value
            lumens = read_light_lumens()

            #Display sensor values on OLED
            oled.text("Temp: {:.1f} F".format(temp_f), 0, 0)
            oled.text("Humidity: {}%".format(hum), 0, 13)
            oled.text("Light: {}".format(lumens), 0, 26)

            #Display last operator message if present
            if last_msg_from_op:
                oled.text(last_msg_from_op[:16], 0, 40)
                oled.text(last_msg_from_op[16:32], 0, 53)


            oled.show()

            #Publish sensor data to MQTT topics
            client.publish("temp", "{:.1f}".format(temp_f))
            client.publish("humidity", str(hum))
            client.publish("light", str(lumens))

            print(f"Published temp: {temp_f}, humidity: {hum}, light: {lumens}")
```

Challenges:

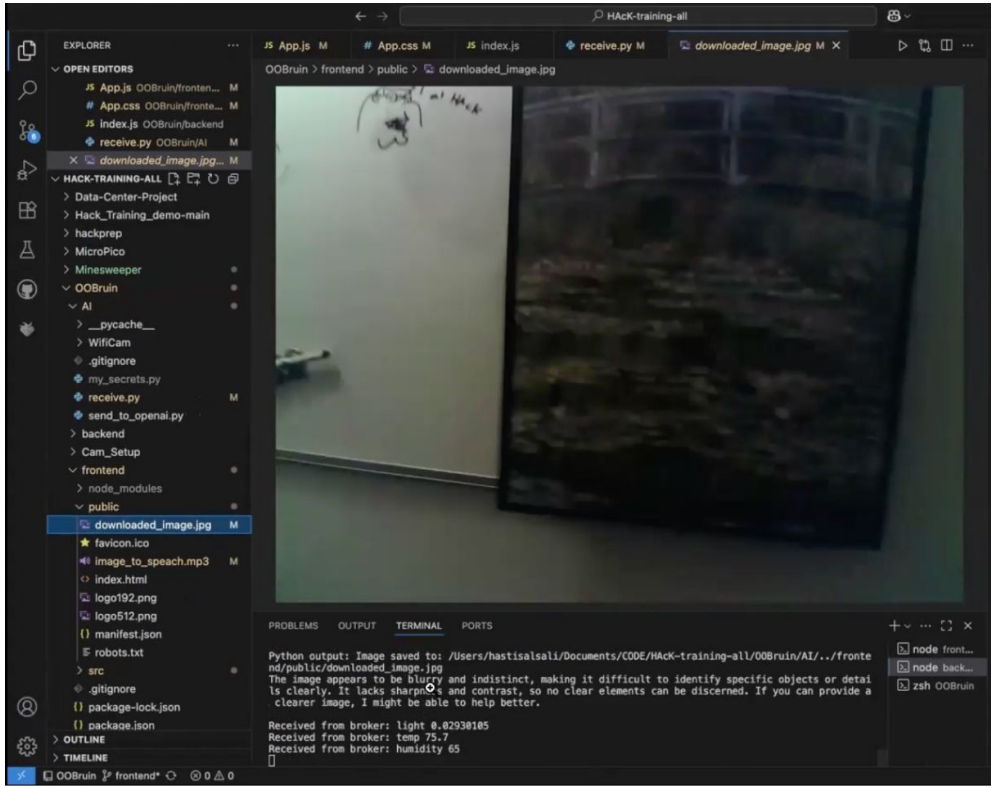
Circuit Fitting Challenges - Edwin



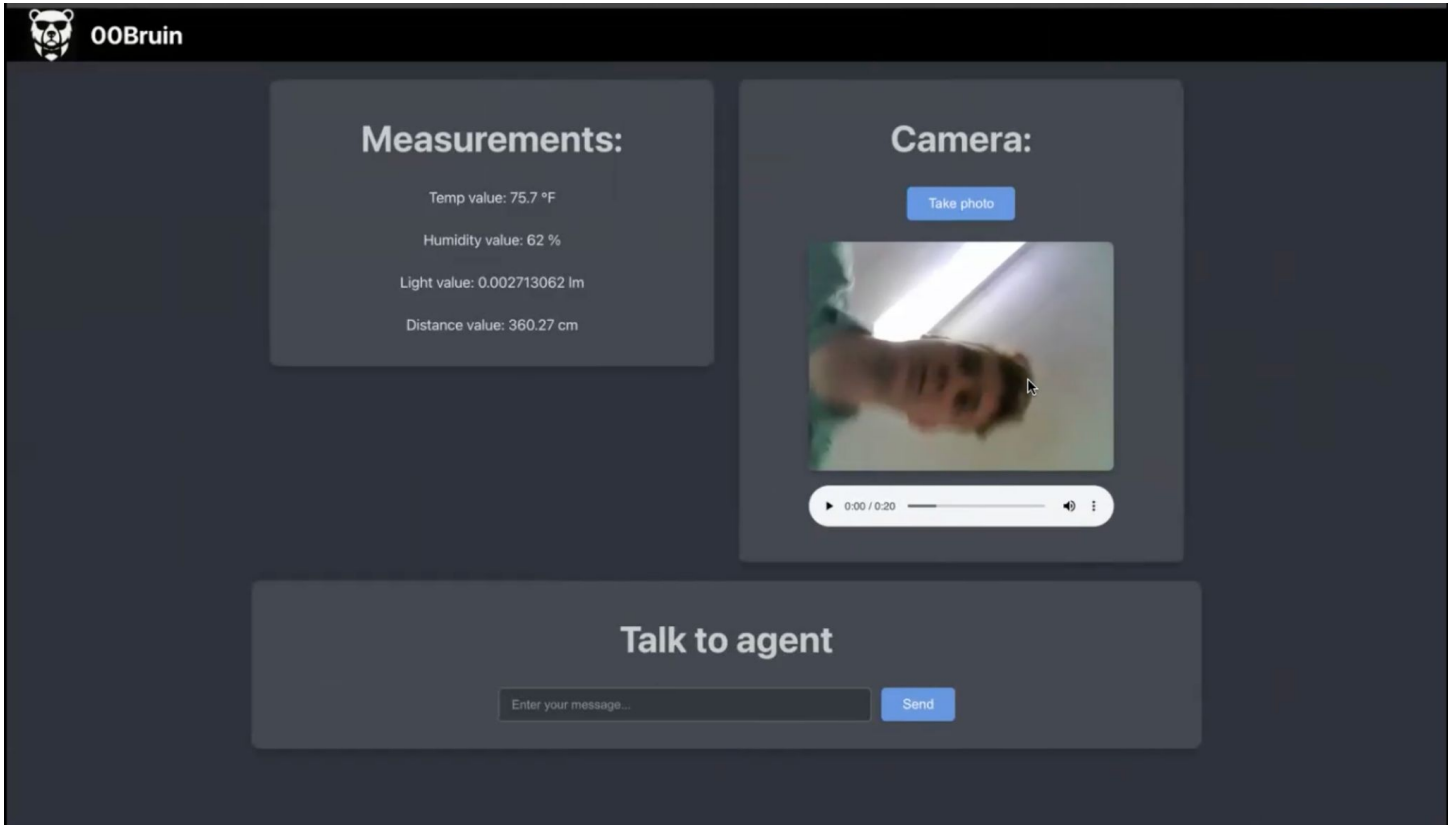
Having a remote member! - Alex

3D Printing Fails - Tyler

AI Photo Detection Issues - Hasti



Ultrasound Failure - Hasti



Reflection/Lessons Learned

- Plan out our project more extensively
- Circuit/component placement
- Time efficiency
    - Would have allowed more time test/debug our spy product.

Future

- Find a way to make the messaging more efficient
- Add a button to switch the watch from a clock to our spy watch
- Make the devices truly separate instead of being wired together

Thank You!