INTERACTIVE MEDIA

DOCUMENTATION WEEK 9

WARNING LIGHTS

VIDEO LINK IN CODE

SYED FAHAD RIZWAN

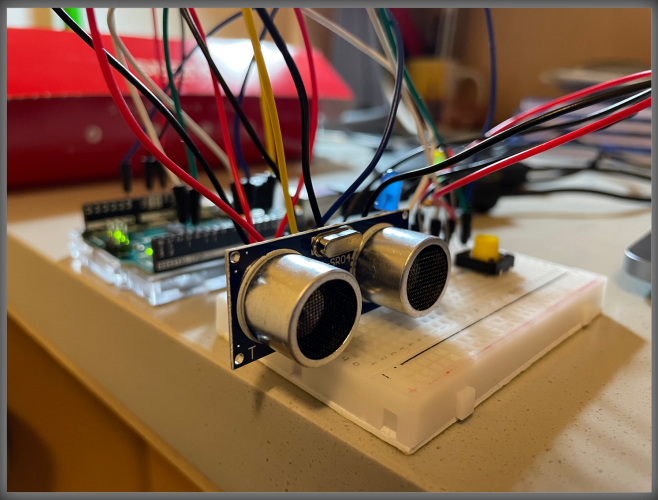
"FEEDBACK IS THE BREAKFAST OF CHAMPIONS"

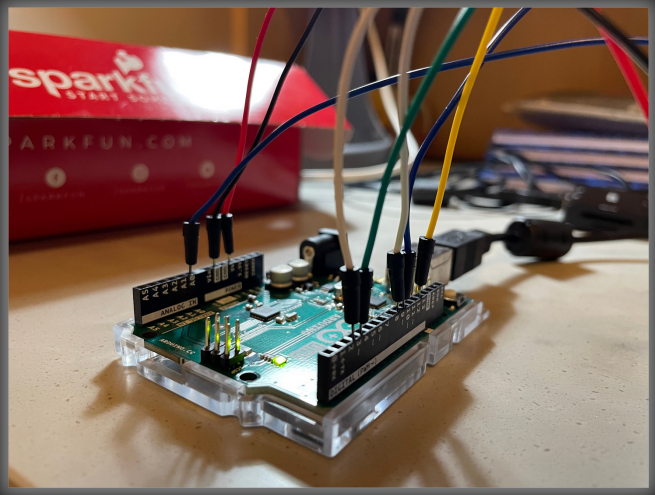
KEN BLANCHARD

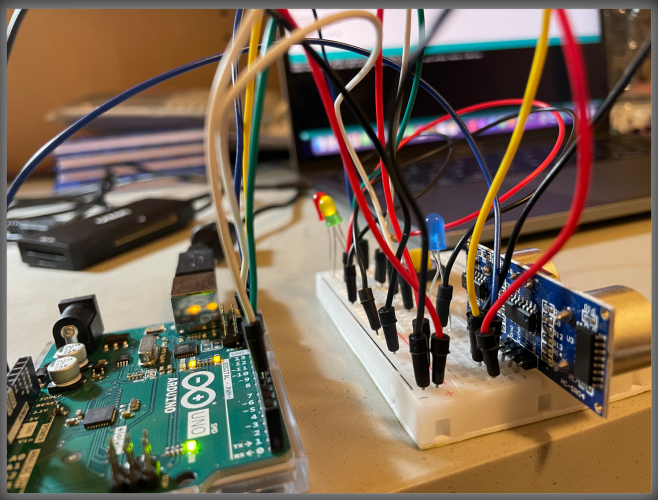
IDEA

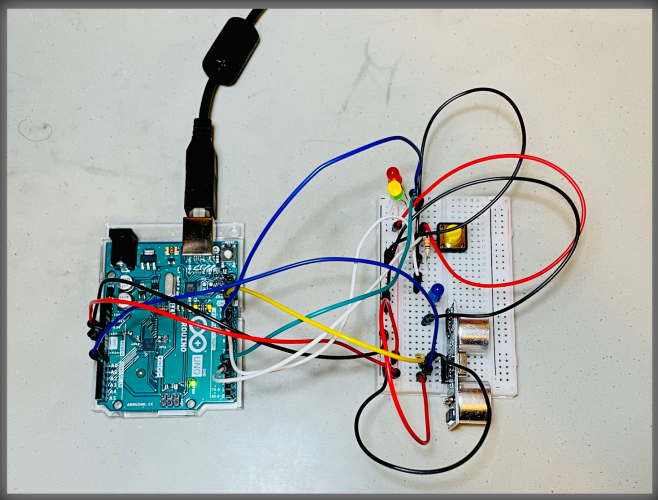
Using an ultrasound sensor to indicate danger levels via colors and intensities of Red and Green LEDs. It is primarily analogue, with the varying sensor output and varying LED intensity.

Additionally, I connected a **Blue** LED to a standard **push button** to demonstrate **digital input and output.**









HOW DOES IT WORK?

ANALOG COMPONENT

The **ultrasound sensor** calculates the distance between the nearest object infront of it. My code uses that distance to exhibit 3 different settings and/or colors of LEDs.

- If **distance is more than or equal to 15cm GREEN LED**
- If **distance is less than or equal to 10cm BRIGHT RED LED**
- If **distance is between 10cm and 15cm DIM RED LED**

POTENTIAL APPLICATION

If an industrialist wishes to ensure that all workers maintain a sufficient distance from the furnaces to avoid heart burns, they could indicate the warning level in different parts via lights.

GREEN could mean safety approved
DIM RED could mean bordering unsafe zone
BRIGHT RED could mean strictly out of bounds

DIGITAL COMPONENT

The **pushbutton** turns on the **BLUE LED** when pressed. It uses **digitalRead()** to **digitalWrite()** on the LED pin, turning it on. It has no direct connection with the analog component yet they coexist on the same circuit. However, a **sliding button** could be used to turn on the device. Using a **pushbutton** eliminates the purpose of automatic safety.

CODE SNIPPETS

CONDITION SETTING

```
// conditions on turning blue LED on or off with pushbutton
if (ButtonValue != 0) {
  analogWrite(Blue_LED, 10);
}
else {
  analogWrite(Blue_LED, 0);
}

// conditions on turning red and green LED on or off via distance
if (distance <= 10) {
  digitalWrite(Red_LED, HIGH);
  digitalWrite(Green_LED, LOW);
}
else if (distance > 10 && distance < 15) {
  analogWrite(Red_LED, 1);
  digitalWrite(Green_LED, LOW);
}
else if (distance >= 15) {
  digitalWrite(Green_LED, HIGH);
  digitalWrite(Red_LED, LOW);
}
```

CRUX

```
// loop function
void loop() {
  // loops trigPin on HIGH state for 1000 micro seconds before LOW
  digitalWrite(trig, HIGH);
  delayMicroseconds(1000);
  digitalWrite(trig, LOW);

  // digitally reads button input (0 or 1)
  ButtonValue = digitalRead(ButtonPin);

  // Reads echoPin, returning sound wave travel time in microseconds
  duration = pulseIn(echo, HIGH);

  // Calculates distance in centimeters
  distance = (duration / 2) / 28.5;

  // Prints distance on serial monitor
  Serial.println(distance);
```

While my code explicitly mentions **digitalRead(), digitalWrite(), and analogWrite(),** it does not mention **analogRead()** yet I believe the echo pin - connected to **A0** - is read via pulseIn, which is essentially the same.

The **Serial.println** does not impact the code. I merely printed it to completely understand the functionality of the ultrasound sensor.

CREDITS / INSPIRATION

Alongside class notes and discussions, I understood the complete functionality of the **ultrasound sensor** and some associated parts like 'println', 'pulseIn', 'echo'. and 'trig' via arduino discussion platforms and YouTube.

CHALLENGES & SOLUTIONS

Hardware is more challenging for me compared to software, primarily as it takes more effort to fix and debug. **Tinkercad** has been a huge help. I plan to keep on playing to get more comfortable. For now, even pushbuttons freak me out.

Additionally, I had a hard time ensuring software ran in synchrony with hardware and remembering the specific applications of PWM. I will now take more notes in class as its useful to quickly have such information at your disposal to not detract from your main idea.

Other than that, the experience was fun!