Exercise 3: basic i/o and iot

Question I started with

How can the experience of playing music in the car be altered to feel more embodied for a driver?

Inputs

(1) The first input sensor I used the **ultrasonic sensor** to capture how far away the nearest object was. In this case, the sensor would be placed near the gear shift and face towards the driver side to be able to sense **whether the driver was in the driver's seat.**

(2) The second input sensor I used was a **button**. This simply captured the data of **whether the button was or was not pressed.**

Outputs

Originally, I had (3) outputs.

(1) I used **webhooks/Spotify** with IFTTT to pause the music when the driver left the seat.
(2) I used **webhooks/Spotify** with IFTTT to play the music when the driver entered the seat.
(3) I used **webhooks/Spotify** with IFTTT to add the current song to a playlist when the button is pressed.

I added a fourth webhook as I realized I did not use a *different* second type of output. This changed my question to be a little broader, as I started to wonder how I could utilize the input data of whether the driver was in the driver's seat to remind the driver of something they needed to do when they got to their destination.

(4) I used **webhooks/notifications** with IFTTT to send a notification to the driver's phone when they left the seat to remind them that they arrived at their destination and to text anyone they might have promised they would text when they arrived.

UX impact + More questions to answer with Arduino prototype

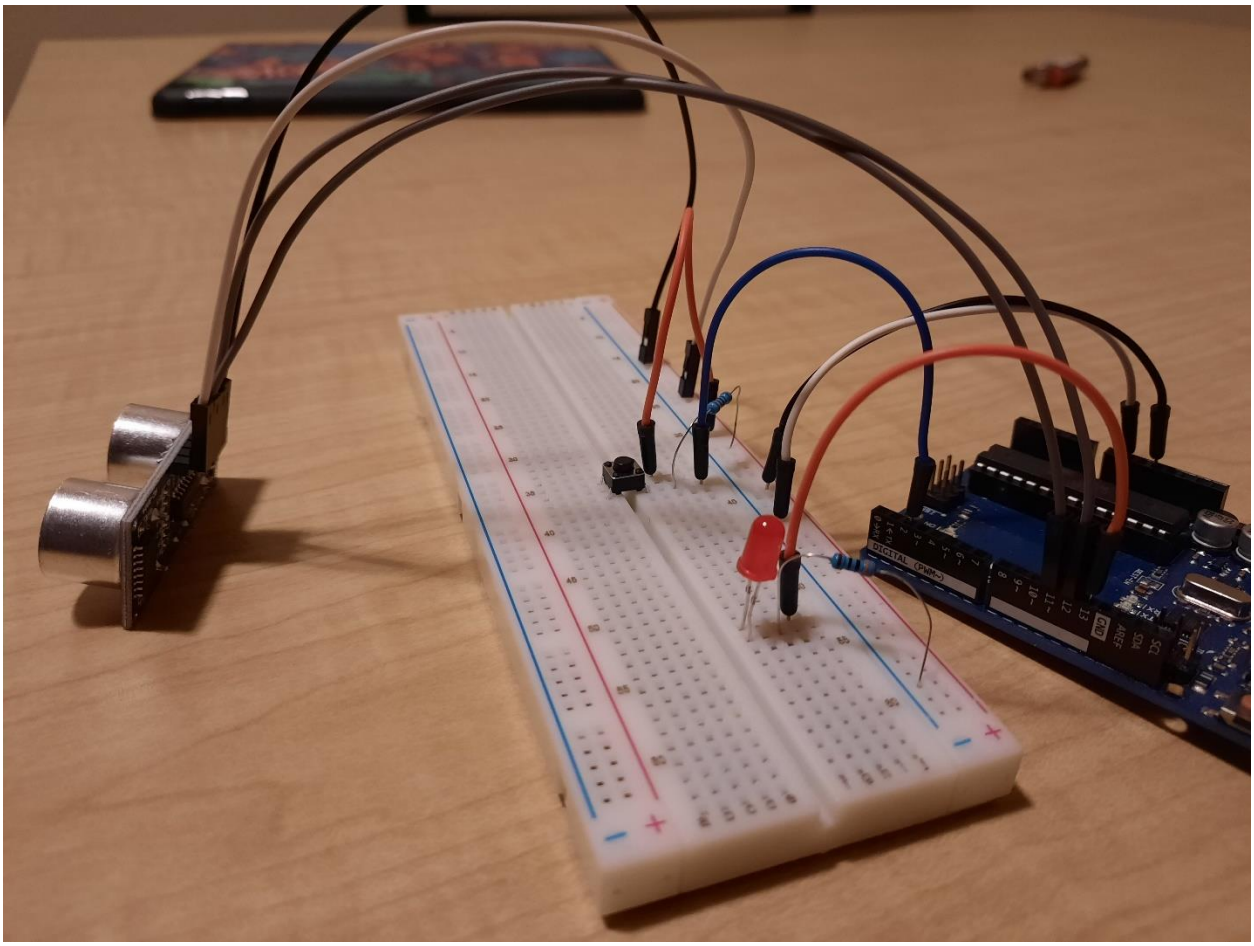
The first two outputs were what really answered my question of embodiment for a driver in regard to music. For a little backstory, a few weeks ago I went to work for eight hours and accidently left my phone attached to the aux in my car. It was awesome that the music played immediately when I got in without having to take the time to plug in the aux, open Spotify, choose music, etc., but it was not so awesome that my phone was nearly dead because it had been playing the entire shift. For this project, **I wondered how I could prototype that experience** without the music playing the entire time.

The third output is something that has always bothered me. **I wish that I was able to hit a button on the dashboard of my car to add a song to a playlist**, because I never remember after I get to my destination what song it was that I liked. So, I tried to prototype a similar experience using Arduino.

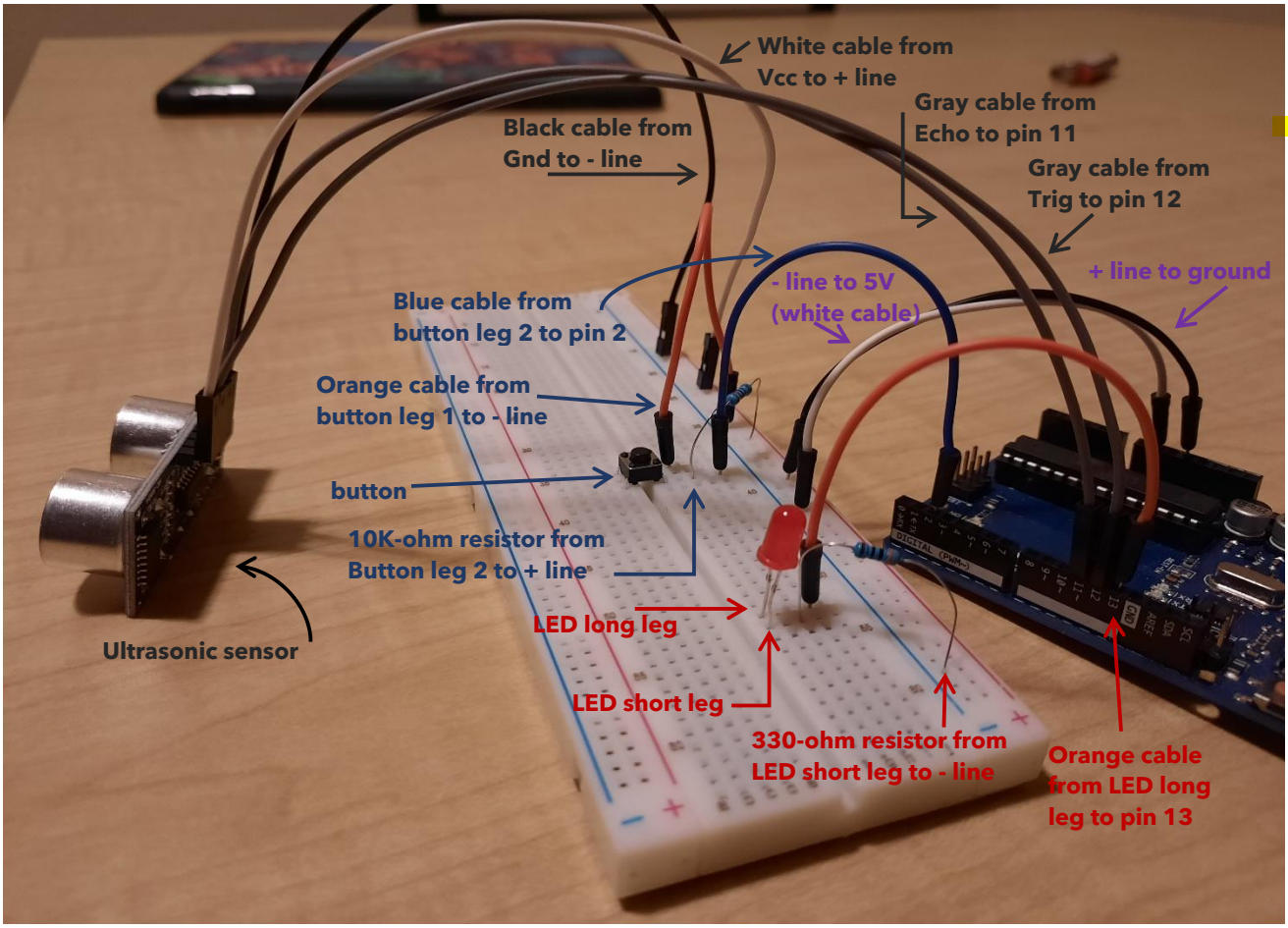
The last output was meant **to prototype a fix to something that I struggled a lot with** in high school. I would tell my mom that I would text her when I got somewhere, but I nearly always forgot to do so, and nearly always got in trouble for it. I wondered how I could use **embodiment** to **remind me to do something when I arrived at a location.**

Arduino setup

This is how I setup my Arduino. I used Austin's provided tutorials to do so. The first picture is provided to make it easier to see individual wires, the second is annotated for clarity.



universal Ultrasonic sensor circuit Button circuit LED circuit



Arduino Code

```
//overall, used button example in Arduino and Austin's in-class tutorial of
ultrasonic sensor

//define ultrasonic sensor stuff
#define TRIGGER 12
#define ECHO 11

// constants to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// button variable to change later
int buttonState = 0;         // variable for reading the pushbutton status

// setup, used Austin's arduino stuff for ultrasonic + button example
void setup()
{
    // LEDpin as output
  pinMode(ledPin, OUTPUT);
```

```
  // buttonPin as input
  pinMode(buttonPin, INPUT);
  // ultrasonic sensor stuff from Austin's example
  pinMode(ECHO, INPUT);
  pinMode(TRIGGER, OUTPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(TRIGGER, LOW);
  Serial.begin(9600);
}

//main loop
void loop()
{
  //this is ultrasonic stuff from example
  long distance, duration;
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  duration = pulseIn(ECHO, HIGH);
  distance = duration / 2 / 7.6;

  //if nearest distance is less than 100 (changed from Austin's example for my
specific experience), print "close" to the serial which processing will then read
to send relevant getrequest
  // the ultrasonic sensor would sometimes read the distance as over 8000 when
something was really close to it, so I added an operator to make sure that wouldn't
happen
  if(distance < 100){ //relational operator & logical operator
    Serial.println("close");
    Serial.println(distance);
   //delay of 5000 so that it would check the distance a little less often (changed
from example to be more relevant to my prototype)
    delay(5000);
  }
  //if nearest distance is more than 100 and less than 8000 print "far" to the
serial which processing will then read to to send relevant getrequest
  else {
    digitalWrite(ledPin, LOW);
    Serial.println("far");
    Serial.println(distance);
    //delay of 5000 so that it would check the distance a little less often
(changed from example to be more relevant to my prototype)
    delay(5000);
  }
  //used the Arudino button example
   // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
    //added this serial print myself which processing will read to know to send
getrequest
```

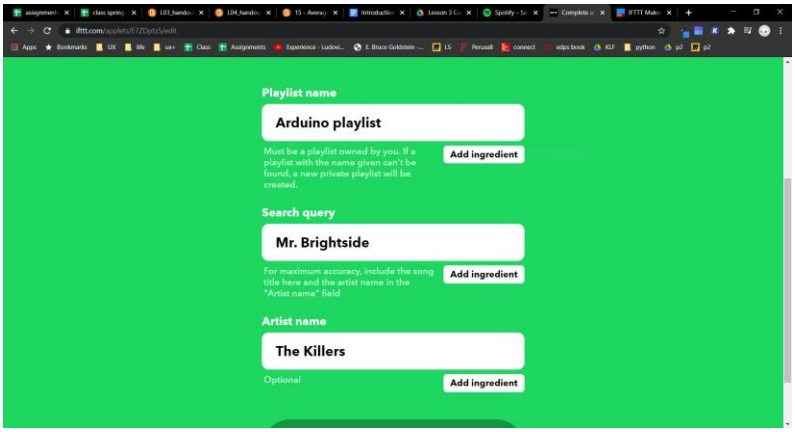
```
    Serial.println("press");
  }  else {
    // turn LED off:
    // added this serial print myself
//    Serial.println("button is not pressed");
    digitalWrite(ledPin, LOW);
  }
}
```

Processing Code

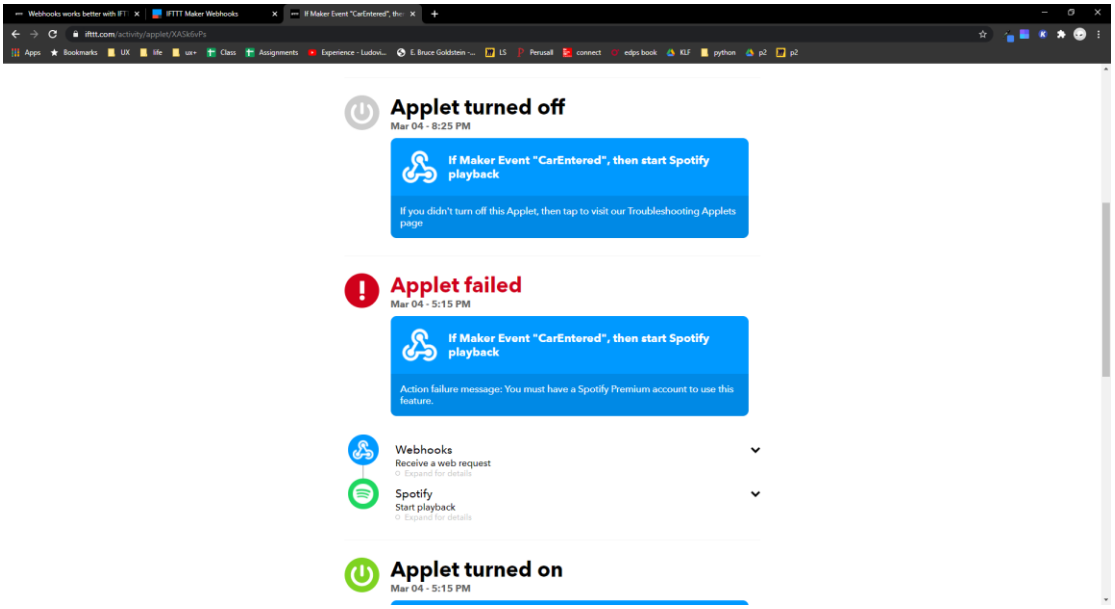
```
//imports + setup
//used Austin's in-class setup for ultrasonic sensor + using IFTTT
import processing.serial.*;
import http.requests.*;
Serial myPort;
String serialValue;

//setup to connect to arduino, used Austin's in-class setup
void setup()
{
  myPort = new Serial(this, "COM5", 9600);
}

//main loop, overall used Austin's in-class setup
void draw()
{
  if (myPort.available() > 0)
  {
//serialValue variable is the string that prints in myport
    serialValue = myPort.readString();
//if the ultrasonic sensor reads that the driver is in the seat: print heard the
word close, getrequest to start spotify using webhooks IFTTT
      if (serialValue.contains("close")) {
        println("heard the word close");
//this works because playing spotify will not work if the instance of spotify is
already playing
      GetRequest get = new
GetRequest("https://maker.ifttt.com/trigger/CarEntered/with/key/dQrdTHAQGS7PO_0Ibyd
v7p");
      get.send();
//if the ultrasonic sensor reads that the driver is not in the seat: print far,
getrequest to stop spotify playback using webhooks IFTTT
//this works because the webhooks will not work if the instance of spotify is
already paused
//also, it will do a second getrequest to send notification through IFTTT app to
remind driver
    } else if (serialValue.contains("far")) {
      GetRequest get = new
GetRequest("https://maker.ifttt.com/trigger/CarExited/with/key/dQrdTHAQGS7PO_0Ibydv
7p");
      get.send();
//I added this getrequest myself
```

```
      GetRequest get2 = new
GetRequest("https://maker.ifttt.com/trigger/Arrived/with/key/dQrdTHAQGS7PO_0Ibydv7p
");
      get2.send();
//if the button is pressed: print "heard added to playlist", getrequest to add song
to playlist
//I added this section myself to add getrequest in response to button
    } if (serialValue.contains("added to playlist")) {
      println("heard added to playlist");
      GetRequest get = new
GetRequest("https://maker.ifttt.com/trigger/buttonpressed/with/key/dLeGyzE2qXKdK6sX
ZRxGdK");
      get.send();
//delay so that it doesn't try to do it alot of times
      //delay(100000);
//if the button is not pressed: do nothing
//I added this myself to make sure that if the button is not pressed nothing
happens
    } else if (serialValue.contains("button is not pressed")) {
    }
  }
  background(0);
}
```

A few struggles & the Overcoming



Originally, I wanted the song added to the playlist to be the one currently playing. Unfortunately, IFTTT with Spotify does not have that capability. **I faked that experience using my prototype** by **specifying which song is added to the playlist** with IFTTT's built in parameters, and then demonstrating the button press **while that specific song was playing.**



I also had an issue with IFTTT not allowing me to use the pause/play Spotify connection without a premium account. I overcame this by using my partner's account, but it was very much a struggle. I ended up having to create five IFTTT accounts, because they wouldn't let me delete old applets and many were automatically connecting to my own Spotify account, and only utilizing two of those accounts' applets.

Entire prototyped experience

With my prototyped experience, demonstrated with a video here, when a driver enters the car, the music will automatically start playing. When a song comes on that the driver wants to add to a playlist, they can simply hit the button and it will be added. Lastly, when the driver exits the car, the music will automatically stop playing and the driver will receive a notification to their phone reminding them to text someone if they need to.

Reflection

I had a lot of fun prototyping this experience with Arduino. **At times it was extremely frustrating**: Internet was not reliable while filming my video outside, my Arduino just quit unexpectedly and would not respond to power source for a short time, and **I didn't know how to do everything I wish I knew how to do** (mostly because of time). For instance, my final experience sends many push notifications through IFTTT (one every five seconds that it reads "far") because I needed it to check that often for the sake of the Spotify pausing within a small amount of time. **I know I could've eventually figured that out**, but that is something I would've done if I had had more time and it was supposed to be a more finalized experience. **I really enjoyed the "quick-and-dirty" mindset I had to take** in regard to prototyping- I have spent so much time thinking a prototype needed to be perfect and I am actually very happy to turn in **a prototype that portrays the experience accurately without working flawlessly**. I think that mindset is something very valuable to take with me as I continue in UX. At this point, **I still wish I knew more about Arduino than I do**. However, I think being able to recognize it as a prototyping tool that can do more advanced & networked experience prototyping is very useful and may come in extremely handy one day.