PROJECT FINAL: WEARABLE

David Snow

Wearable Computing
Jackson McConnell
Dec/09/2014

DEVICE: 'iRIS'

Project iRIS is a device that gamifies the experience of your daily drive. This is a device for all those drivers who have a boring driving experience everyday, in its basic form iRIS is a timer that provides a race like start at the beginning of your drive to your destination.

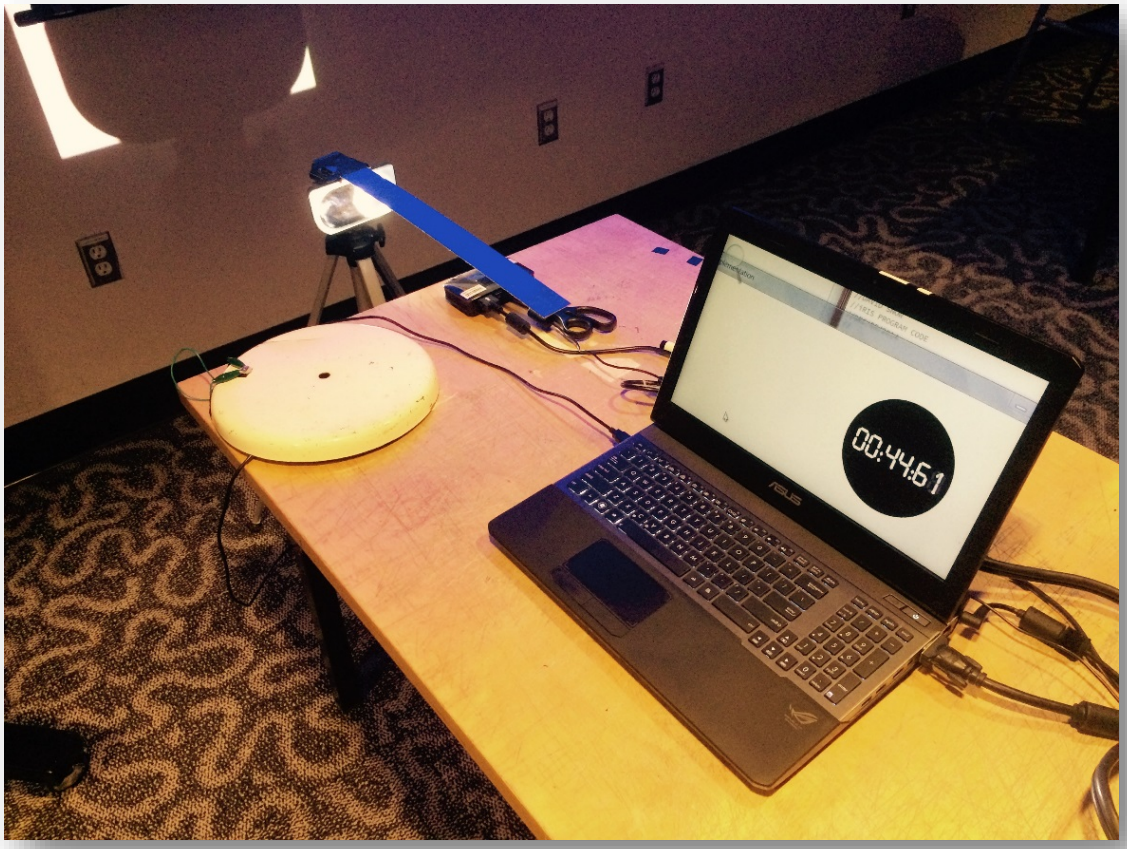
Basically what this device is intended to do is create a fun experience of getting to your work everyday, which after a while becomes mundane. The race like timer displays a countdown from 5 when the user is ready, at the moment it reaches `GO!` a timer begins counting upward. The point of this timer is so that once the user reaches their destination, they are ablet to stop the timer and compare to their previously recorded times, to see if they are becoming more efficient.

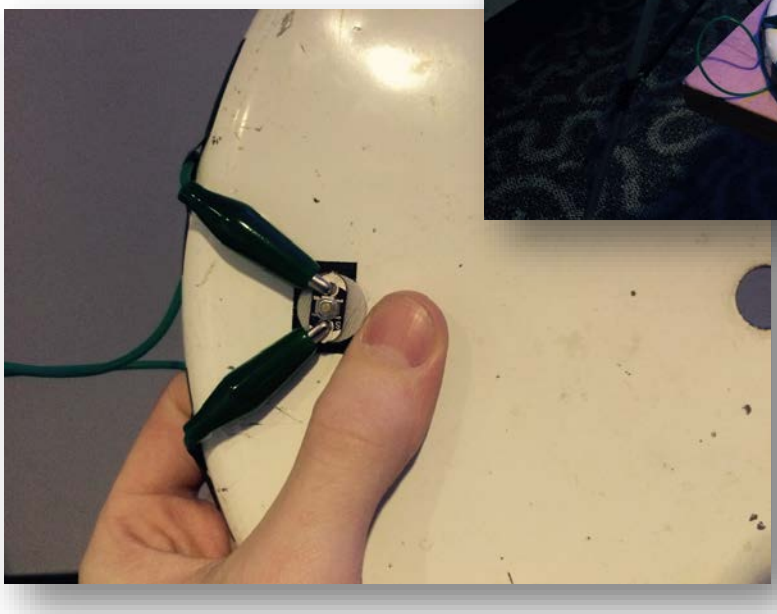
What I wanted to do with this project is create a stress level on top of the daily drive that reminds them that it is possible to have fun with driving, especially when people take it very muich for granted. My vision consisted of using a race like start, a timer function, and buttons to create an expereince that would rival on a track.

Wearable Setup

The following is the setup I created for my inclass project. It consists of a projector, a computer, an arduino with a button attached, as well as a mirror to project onto.

In this project I demostrated that this would actually be a part of the vehicle and would be mounted on the ceiling of the vehicle itself, while projected downward towards the mirror where it will be displayed. A thing vinyl would be needed to be layered onto the mirror in order to properly have an animation running into the mirror, or else it would just be projecting into the eyes of everyone behind. The following are images of the count down sequence as well as the general setup.





This is what is actually displayed on the mirror itself, it includes a READY function, and once started counts down from 5 until GO. This is what getst the user all ready, I believe that it was efficient in displaying the type of level and intensity that was required for the project.

The following is all the arduino as well as processing code that I spent many an hours working on, I had a lot of fun this time around playing with the code for my project, and luckily enough didn't run into any crazy problems.

Arduino Code

```
//DAVID SNOW
//iRIS PROGRAM CODE
//DEC/05/2014
//
//WEARABLE COMPUTING PROJECT
//JACKSON MCCONNEL

int buttonOne = 6;                    //Variable for pin 6
int buttonValue;                      //Variable for button value

//Setup function before code is run
void setup(){
  Serial.begin(9600);                 //Bodrate for communication

  pinMode(buttonOne, INPUT);          // Sets up pin 6 as button
  digitalWrite(buttonOne, HIGH);      // Ties button to high
}

//Loop function for running and reading
void loop(){
  buttonValue = digitalRead(buttonOne);   //Read button value and store
  if(buttonValue == LOW){               //if button has been pressed then continue

  Serial.println("press");              //Send "press" over serial
  delay(1000);                        //Delay for 1000 mille for ease
  }
}
```

Processing Code

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
//DAVID SNOW                                                    +
//iRIS PROGRAM CODE                                                 +
//DEC/05/2014                                                 +
//                                                        +
//WEARABLE COMPUTING PROJECT                                                +
//JACKSON MCCONNEL                                                  +
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
```

```
import processing.serial.*;                    //Import processing serial communication library
import ddf.minim.*;                            //Import minim music library

//GLOBAL VARIABLES
//-------------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------------

//Serial communication variables
Serial myPort;
String communication;                          //Variable to store what has been read
boolean commUse;                               //Boolean used for knowing when arduino has communicated

//Radius of circles
Float radiusOuter = 208.0;                     //Radius size of circles
Float radiusInner = 208.0;

//Circle Coordinates
int X, Y;                                  //Variables for coordinates of assets

int commUseCounter = 0;                        //Creates counter for pause and restart function

//Counter string number
int countDown;                                 //Number used to display countdown from 5
int milleTimer;                               //Timer variable that stores mille laps
int countUpMin, countUpSec, countUpMil;              //Variables used to store timer slots

//Strings for startup and ending words
String textAll;                               //Variable to store text displayed

//String for the timer function
String minute, second, millesecond;                 //String variables for timer slots

//Boolean variable to tell timer to start
Boolean go;                                   //Boolean for whether GO has been displayed

//Create var for font
PFont segmentDisplay;                          //Variable to store font of 7 segment

//Create audio
Minim minim;
AudioPlayer soundOne;                          //Low tone that happens 4 times on count down
AudioPlayer soundTwo;                          //High tone that happens after 4 low tones


//SETUP FUNCTION
//-------------------------------------------------------------------------------------------------
//Sets up all variables before program begins.
//-------------------------------------------------------------------------------------------------
void setup () {

  //Port communication between programs
  String portName = Serial.list()[0];              //Set variable for port reading
  myPort = new Serial(this, portName, 9600);          //set port for reading and bodrate

  commUse = false;                             //Set boolean communication check to false
```

```
  size(900, 300);                          //Set size to full screen
 frameRate (35);                           //Set framerate of program to 35fps

 //Load sounds into program
 minim = new Minim(this);                  //Create new minim
 soundOne = minim.loadFile("Low_Tone.mp3");        //Load low tone sound from data file
 soundTwo = minim.loadFile("High_Tone.mp3");       //Load high tone sound from data file

 segmentDisplay = createFont("LiquidCrystal-Bold", 48);//Load 7 segment font into sketch

 //Set X and Y coordinates of circle
 X = width /2 + 200;                       //Set X coordinate to half screen size plus "
 Y = height /2;                           //Set Y coordinate to half height of screen

 countDown = 5;                            //Set countDown number to start at 5

  //Set the timer numbers before they count
 milleTimer = 0;                           //Set all to 0
 countUpMil = 0;
 countUpSec = 0;
 countUpMin = 0;

 //Sets string variables for timer
 minute = "0";                            //Start minute at 0
 second = "0";                            //Start Second at 0
 millesecond = "0";                        //Start millesecond at 0

 go = false;                              //Go to false because button hasnt been pressed

 textAll = "READY";                         //Set string text to "ready" for beginning

 background(255);                          //Set background color of program
}

//DRAW FUNCTION
//-----------------------------------------------------------------------------------------------
//Runs through all following code over and over until program is terminated
//-----------------------------------------------------------------------------------------------
void draw () {

  //If port is available to communicate then..
  if (myPort.available() > 0) {
    communication = myPort.readStringUntil('\n');       //Set communication variable to whatever is recieved
    commUseCounter += 1;                        //Increases comm counter by 1 each time entered

    commUse = true;                           //If communication is sent then commUse is true
  }

  //Check for if GO has been executed as well as not entered into port yet..
  if (go == false && commUseCounter == 0) {

    background(255);                          //Set background color of program

    //Call class containigCircle
    containingCircle();
```

```
  //Call class containigText
  containingText();                          //Display "Ready" on screen

  delay(300);
}

//Check if a communication has been sent from Arduino
if (commUse == true && commUseCounter == 1) {

  background(255);                           //Set background color of program

  //Call class containigCircle
  containingCircle();

  //If statements controlling whether or not tones are sounded
  if (radiusOuter == 208 && countDown > 0) {        //If at beginning of each number sequence
    soundOne.play();                         //Play low tone sound
    soundOne.cue(0);                         //Get ready for next
  } else if (radiusOuter == 208 && countDown == 0) {//If 4..3..2..1 has been done then next
    soundTwo.play();                         //Play high tone sound
    soundTwo.cue(0);                         //Get ready for next
  }

  //Call class countdownCircle
  countdownCircle();

  //Call class containigCircle
  containingCircle();

  //Call class count down numbers + "GO"
  containingCountDown();
}

//Sets the program into a pause state
if (commUseCounter == 3) {
  go = false;
  delay(300);
}

//Sets the program into a reset state
if (commUseCounter >= 4) {
  commUse = false;
  commUseCounter = 0;

  //Set the timer numbers before they count
  milleTimer = 0;                            //Set all to 0
  countUpMil = 0;
  countUpSec = 0;
  countUpMin = 0;
}

//When the timer is ready to start counting up
if (go == true && commUseCounter >= 1 && commUse == true) {

  background(255);                           //Set background color of program
```

```
   //Call class containigCircle
   containingCircle();

   //Timer that displays counting up after 5..4..3..2..1..GO
   containingCountUp();

   //Set comm counter to next stage to not enter back into count down
   commUseCounter = 2;
  }
}

//CONTAINING CIRCLE FUNCTION
//-----------------------------------------------------------------------------------------------
//Draws a black circle that is displayed that text and other numbers sit on top of
//-----------------------------------------------------------------------------------------------
void containingCircle() {

  fill (0);                                  //Set fill color
  strokeWeight(8);                           //Set size of stroke weight
  stroke(255, 255, 255);                     //Set color of stroke to white
  ellipse ( X, Y, radiusInner, radiusInner);          //Draw ellipse or circle
}

//OUTER CIRCLE FUNCTION
//-----------------------------------------------------------------------------------------------
//Circle that increases outside black circle to visualize a count down
//-----------------------------------------------------------------------------------------------
void countdownCircle() {

  radiusOuter = radiusOuter + 2;                  //Increase size of out circle by 2 each time

  //Resets circle to origional size once at largest
  if (radiusOuter >= 300.0) {
    radiusOuter = 208.0;                          //Reset outer circle radius
    countDown = countDown - 1;                    //Decrease count by 1 each time for count down

    //Resets text to origional once at 0
    if (countDown == -1) {
      //commUse = false;                          //Sets boolean variable for comm to false
      countDown = 5;                            //Reset countdown number to 5
      go = true;                                //Set GO variable to true to say it has begun
    }
  }

  strokeWeight(4);                              //Set stroke weight to 4
  fill (255, 0, 0);                             //Fill color to red for outer circle

  //Changes color of circle for final blurp
  if (countDown == 0) {
    fill (0, 255, 0);                           //Fill color to green for outer circle
  }

  ellipse ( X, Y, radiusOuter, radiusOuter);          //Draw ellipse for outer countdown animation
}

//INNER TEXT FUNCTION
```

```
//-------------------------------------------------------------------------------------------------
//Displays the 5..4..3..2..1 as well as GO inside of the circle when pressed
//-------------------------------------------------------------------------------------------------
void containingCountDown() {

  textFont(segmentDisplay);                    //Sets the font to 7 segment font
  textSize(120);                               //Sets size of font to 120
  textAlign(CENTER, CENTER);                   //Align text to center of location
  fill(255, 0, 0);                             //Fill the color to RED
  text(countDown, X + 9, Y);                   //Display text of numbers on screen

  //If countdown is at 0 say "GO!"
  if (countDown == 0) {

    //Call class containigCircle
    containingCircle();

    textFont(segmentDisplay);                  //Sets the font to 7 segment font
    textSize(100);                             //Sets size of font to 100
    textAlign(CENTER, CENTER);                 //Align text to center of location
    fill(0, 255, 0);                           //Fill the color to GREEN
    text("GO!", X + 5, Y);                     //Display GO on screen once counted down
  }
}

//STARTUP AND END TEXT FUNCTION
//-------------------------------------------------------------------------------------------------
//Displays text on screen before and after real function of program has begun
//-------------------------------------------------------------------------------------------------
void containingText() {

  textFont(segmentDisplay);                    //Sets the font to 7 segment font
  textSize(72);                                //Sets size of font to 72
  textAlign(CENTER, CENTER);                   //Align text to center of location
  fill(255, 255, 255);                         //Fill the color to white
  text(textAll, X + 5, Y);                     //Display textAll variable
}

//COUNTING FUNCTION
//-------------------------------------------------------------------------------------------------
//Does math for the timer counting up displayed on screen after the count down process
//-------------------------------------------------------------------------------------------------
void containingCountUp() {

  milleTimer = millis();                       //Sets variable to amount of milleseconds done

  //Counts the milleseconds up
  if (milleTimer >= 1) {
    countUpMil += 3;
    milleTimer = 1;
  }

  //Counts the Seconds up after milleseconds have reached 99
  if (countUpMil >= 99) {
    countUpSec += 1;
    countUpMil = 1;
```

```
  }

  //Counts the minutes up after the seconds have reached 60
  if (countUpSec >= 60) {
    countUpMin += 1;
    countUpSec = 1;
  }

  millesecond = ""+countUpMil;                  //Sets integar variable to string variable
  second = ""+countUpSec;                      //Sets integar variable to string variable
  minute = ""+countUpMin;                       //Sets integar variable to string variable

  //if single digit add 0 in front
  if (countUpMil <= 9) {
    millesecond = "0" + millesecond;
  }

  //if single digit add 0 in front
  if (countUpSec <= 9) {
    second = "0" + second;
  }

  //if single digit add 0 in front
  if (countUpMin <= 9) {
    minute = "0" + minute;
  }

  textSize(50);                                 //Sets size of font to 50
  textAlign(CENTER, CENTER);                        //Align text to center
  fill(255, 255, 255);                          //Set color to white
  text(minute + ":" + second + ":" + millesecond, X + 7, Y);//Display timer on screen
}
```

Parts list

1x LilyPad Simple Board - MCP73831/2
1x LilyPad Button
1x 110mAh LiPo Battery
1x pico projector