Jane Folliard, janette2
Omar Taha, otaha2
Brian Vickers, bpv2
ECE 120/110 Honors Lab

Concussion Sensing Headband

Introduction

Statement of purpose

Concussion awareness in high-impact sports such as football has increased drastically over the last decade, but other sports, like soccer and rugby have not seen the same rise in awareness. High level head impacts that could lead to brain injury still occur in these other sports. Our headband would provide a way to monitor the impacts received by players in sports where players do not wear helmets. Concussion awareness is important to us because multiple group members have sustained concussions from sports other than football. Top football helmet manufacturers like Riddell have started to create helmets that will monitor head impacts, but this technology is not as prevalent in other sports. Our project would be versatile enough that is could be used by players in any sport.

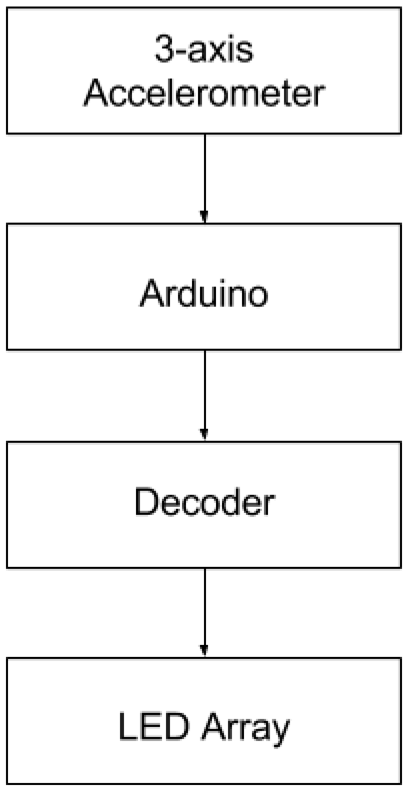
The threshold impact to cause a concussion is still heavily debated and under observation, but it seems any amount of force above roughly 40g can be dangerous. For this reason, we will use a high-g 3-axis accelerometer to measure severe changes in velocity. The primary goal of this project is to create a new way to monitor impacts and provide a way to alert training staff of potential traumatic brain injuries.

Features and Benefits

In order to monitor head impacts and alert training staff of potential traumatic brain injuries, we have created prototype that can monitor the accelerations of such impacts, process the data, and signal when a certain threshold of force has been surpassed. This project is able to monitor drastic changes in velocity using a 3-axis accelerometer. The data is then processed by an Arduino. The Arduino outputs signals to a decoder which controls an LED array. Depending on the level of force sustained, one of the LEDs will light up. Each successive LED signifies a greater amount of force sustained, with adjustable thresholds. The LED indicating the level of impact sustained will stay lit until a greater amount of impact is detected, at which point that corresponding LED will light up and stay lit. Allowing the LEDs to stay lit makes identifying the greatest impact that has occurred easier.

Design

System Overview



Design Details

The 3-axis accelerometer collects the acceleration data. It then transmits this data to the Arduino as an analog signal. The accelerometer measures accelerations from -200g to +200g and outputs analog signals from 0 to 1023. 0 corresponds to -200g, 511 corresponds to 0g, and 1023 corresponds to +200g.

The arduino takes in this information and processes it. It reads the analog signal and translates it to the corresponding acceleration for each axis. The Arduino will then calculate the magnitude of the acceleration and check to see if it passes one of the preset thresholds. If a threshold is passed, the Arduino will change the signals it outputs to the decoder to light the correct LED.

The decoder simply translates a two bit signal from the Arduino to one of four states for the LED array. One output of a decoder is always 1 and the rest are zero.

The LED array consists of four LEDs. Each one is connected to one output of the decoder so that only one will be lit up at a time. The top LED corresponds to the lowest threshold and the bottom LED corresponds to the highest threshold.

Results

We successfully connected the accelerometer to the Arduino using a bundle of wires. We implemented code that would process the gradient output of the accelerometer and convert the data into the amount of G-force experienced. In addition, we built our own decoder using AND and NOT gates and implemented the output logic successfully. In order to test our output logic, we changed the parameters that set the threshold accelerations to trigger an LED to illuminate. Instead of requiring 40g, 60g, and 80g to trigger action from our circuit, we lowered the thresholds to reasonable values like 1g, 2, and 3g. We chose these values because they are low enough that they can be easily simulated by us shaking or dropping the accelerometer. These parameters are the constants ‘first’, ‘second’, and ‘third’ in our code (see appendix). We assume that altering these constants will not cause our project to fail, it will just have altered functionality.

Problems and Challenges

Overall, the major challenge we faced is that none of us had any prior experience in computing, electronics, or design. At the beginning of the semester, our original idea was to use bluetooth to transmit the data wirelessly. After more research, it turned out that Bluetooth unrealistic for the small amount of data we were transmitting. It also became clear that since we were all so new to design and electronics, we should scale down our first semester goals and focus on data collection and processing instead of data transmission. Due to this, we decided to use our time this semester to create a proof of concept. We weren't concerned as much with size as we were with functionality. The breadboard with the accelerometer is not small enough to be realistically worn while playing sports, but it allowed us to learn how to wire an accelerometer. Obviously, it is not realistic for athletes to be wired to a processor, but running wires from the accelerometer to the Arduino allowed us to learn how to process data with an Arduino without struggling with wireless transmission.

Future Plans

Our future plans consist of two main goals, implementing a wireless connection and making the headband component into a PCB. The headband will be a self contained device, potentially with LEDs on the headband to signal the amount of force sustained. A PCB is more functional than the current breadboard design. It is difficult to use the accelerometer on a breadboard because moving it around at high speeds risks causing damage or detaching elements. Alternatively, implementing a wireless connection would allow the data to be transmitted from the wearer to a microprocessor such as an arduino. We have also considered including the microprocessor on the headband itself so it is self contained. When making these decisions, we need to consider the overall goal of the project and whether or not we want the alert LEDs to be on the headband, somewhere remote such as with medical staff, or some combination of the two.

References

[1]"Most concussions deliver 95 g's, neuropsychologist says", *ScienceDaily*, 2016. [Online]. Available: https://www.sciencedaily.com/releases/2010/06/100624092526.htm. [Accessed: 26- Sep- 2016].

[2]M. Derewicz, "Where g-force and gray matter meet | endeavors", *Endeavors.unc.edu*, 2016. [Online]. Available: http://endeavors.unc.edu/spr2008/football_concussions.php. [Accessed: 26- Sep- 2016].

Appendix

Code

```
const int groundpin = 18;            // analog input pin 4 -- ground
const int powerpin = 4;              // analog input pin 5 -- voltage

const int xpin = A3;                 // x-axis of the accelerometer
const int ypin = A2;                 // y-axis
const int zpin = A1;                 // z-axis (only on 3-axis models)

const int s1 = 22;                   //output S1 pin
const int s0 = 24;                   //output S0 pin

int state = 0;

double xvol;
double yvol;
double zvol;
double xg;
double yg;
double zg;
double gTot;


const int first = 40;
const int second = 60;
const int third = 80;

const double AR_TO_VOL = 3.3/1023;    //constant used to convert analogRead output to a voltage from 0v to 3.3v

void setup() {
  analogReference(EXTERNAL);      //configure reference voltage (top of input range) to V applies to AREF pin

  Serial.begin(9600);

  //pinMode(groundpin, OUTPUT);
  pinMode(powerpin, OUTPUT);
  //digitalWrite(groundpin, LOW);
  digitalWrite(powerpin, HIGH);

  pinMode(s1, OUTPUT);
  pinMode(s0, OUTPUT);

  digitalWrite(s1, LOW);
  digitalWrite(s0, LOW);


}

void loop() {
  //Get voltages from accelerometer
  xvol = (double) analogRead(xpin) * AR_TO_VOL;
  yvol = (double) analogRead(ypin) * AR_TO_VOL;
  zvol = (double) analogRead(zpin) * AR_TO_VOL;

  //Convert voltages to g, compute g_tot
  xg = voltageToG(xvol);
  yg = voltageToG(yvol);
  zg = voltageToG(zvol);

  gTot = sqrt( xg*xg + yg*yg + zg*zg);

  Serial.print("X value: ");
  Serial.print(xg);
  Serial.println();
  Serial.print("Y value: ");
  Serial.print(yg);
  Serial.println();
  Serial.print("Z value: ");
  Serial.print(zg);
  Serial.println();
  Serial.println("Magnitude of G force: ");
  Serial.print(gTot);
  Serial.println();
  Serial.println();

  //based on gTot, output s0, s1 to decoder to light up led array
  if (first <= gTot && second > gTot && 0 == state) {
    digitalWrite(s1, LOW);
    digitalWrite(s0, HIGH);
    state = 1;
  }
  else if (second <= gTot && third > gTot && 2 > state) {
    digitalWrite(s1, HIGH);
    digitalWrite(s0, LOW);
    state = 2;
  }
  else if (third <= gTot && 3 > state) {
    digitalWrite(s1, HIGH);
    digitalWrite(s0, HIGH);
    state = 3;
  }

  delay(500);
}
//Converts voltage value to Gs
double voltageToG (double voltage) {
  voltage -= 1.65;
  double g = voltage *= 200.0/1.65;
  g -= .2;
  return g;
}
```