Week 1 (August 29-September 7)

Tools used:

Python

* Libraries used:
  + Pandas
  + Numpy
  + DateTime
  + Time
  + Calendar

Elan (Video annotation application)

* Tiers
* Annotations
* Exporting to different formats

PivotHead Camera

Goal:

Objective: To prepare for data synchronization among all sensors (consisting of heartbeat / accelerometer / distance using Bluetooth signals / audio) we must use video input as “ground truth”, a way of labeling the action occurring during certain patterns of data. In order for synchronization to happen, we proposed to align all sensor data using the standard Unix Timestamp. For this week, I must create a Python Script that can format an Elan exported file (HTML) and return multiple CSV files for Unix TimeStamp alignment along with the annotations.

* Learn Elan features:
  + How to create annotations
  + Understanding how to use tiers and for what purposes
  + How to export Elan to a readable csv file (where data analysis can be performed)
* Understand how data synchronization works
* Create Python script that formats exported Elan file

Results:

Elan features:

* Successfully created annotations and used different tiers to designate certain categories of infant distress.
  + What it may be like:
  + Tier 1 – Infant activities
  + Tier 2 – Mother activities
  + Tier 3 – Start and finish time
* Elan does not support Unix Timestamp, nor does it give the date or the hour of the day in which a video is recorded. Solution will be discussed in Python implementation

Python Script:

Purpose: The purpose of this script is to be able to take in an exported Elan HTML table, divide each tier into separate components, propagate the start time of recording forward towards the last annotation per tier, export each of these tiers as CSV files, and allow user input of date and time to offset the local video time and replace each interval with a Unix Timestamp in order for data synchronization to occur with other devices.

* Functions:
  + multipleParse()
    - Indicate all html files that need to be formatted in an array
  + parse(fileName)
    - The main function of this script. It parses through each table, finds the tiers, finds the start and end time of each annotation for each tier, creates a DataFrame unique to each tier, propagates time from start to stop in increments of .1 seconds for each annotation for each tier, takes in user input of date and time, finds the offset according to the “Start” annotation, replaces each local video time with a Unix Timestamp, labels each corresponding Unix Timestamp with the appropriate comment within an annotation
  + createDF(start,end,delta,offset,unixTime,action=””)
    - This helper method performs the actual propagation of time given a start and end time, the increment amount, the offset (from Start tier), the Unix Date/Time, and the annotation label
  + roundTime(time)
    - Takes in time of type dateTime
    - This helper method rounds the times to the nearest tenth of a second
  + stringToTime(arg)
    - Takes in String of format “00:00:00.00”
    - Converts a String to a Time for calculations
  + stringConverter(datet)
    - Takes in DateTime
    - Converts Time to String and drops the excess 0’s behind .1
  + findTiers(filename)
    - Finds all unique tier names and returns an array
  + findBeginning()
    - Finds the “Start” tier and returns its offset
  + computeSecondOffset()
    - Given a time input in String, returns the total amount of seconds
  + unix()
    - Given a time of day and a date, returns the Unix TimeStamp





