Open Source HEG Whitepaper

The how's, what's, and why's.

Joshua Brewster

Abstract:

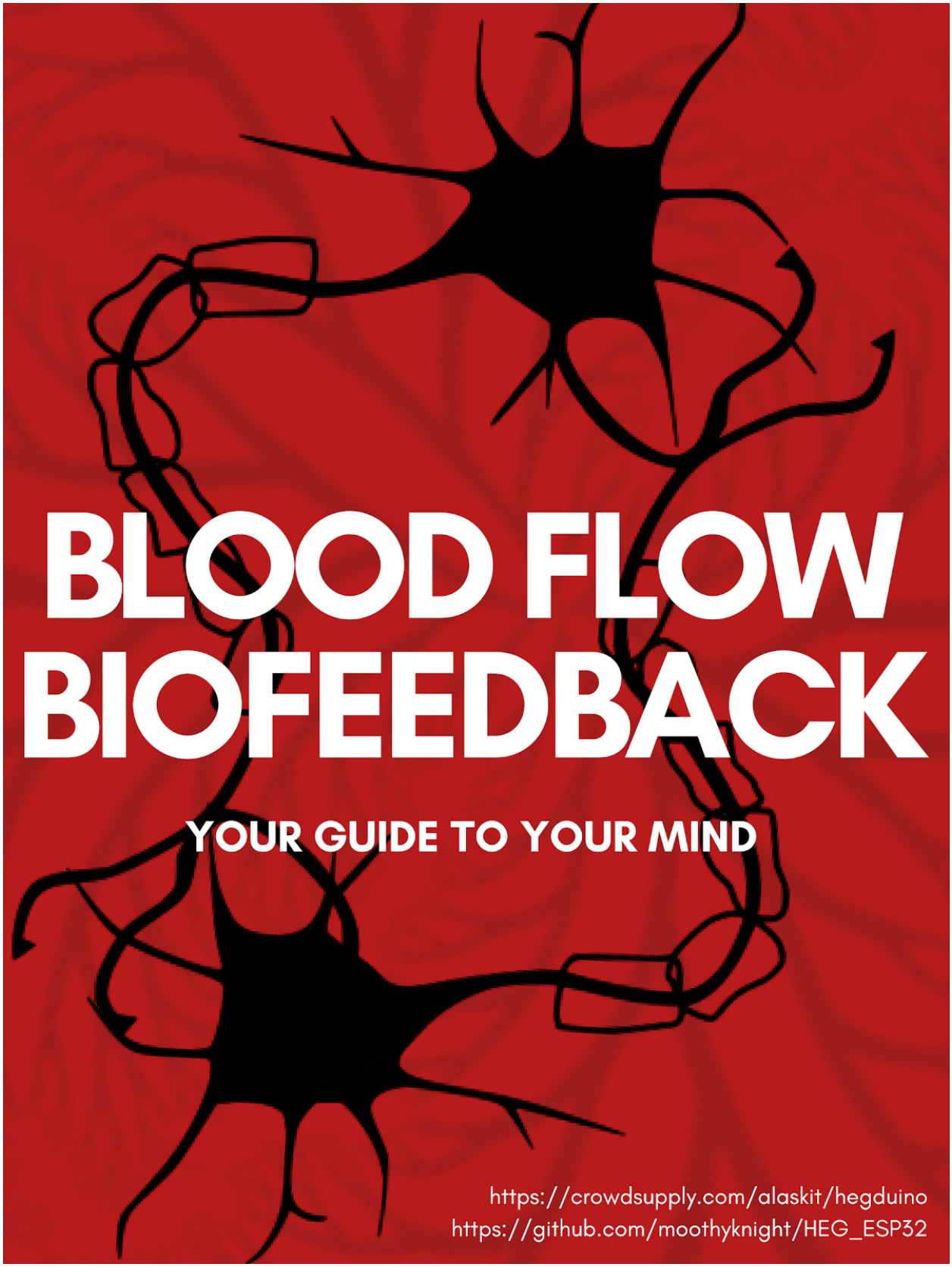
Hemoencephalography utilizes reflective red-infrared pulse oximetry to read and help gain control of blood-oxygen levels in brain regions or other parts of the body. The primary open source HEG model is built with widely used hardware and software solutions with the ESP-32, generic sensors and digital converters, and the Arduino IDE. This serves as a maximally affordable, maximally accessible, and maximally iterable foundation for what is to be seen as a much larger scientific and cultural endeavor. Let this project serve as an archetype of more projects like it, where this whitepaper will establish the desired results of the open source platform and the ethos underlying this attempt. Emphasis is placed on community, creativity, and collaboration.

Contents:

HEMOENCEPHALOGRAPHY

YOUR GUIDE TO YOUR MIND

https://crowdsupply.com/alaskit/hegduino
https://github.com/moothyknight/HEG_ESP32


BLOOD FLOW
BIOFEEDBACK
YOUR GUIDE TO YOUR MIND
https://crowdsupply.com/alaskit/hegduino
https://github.com/moothyknight/HEG_ESP32

Preamble:

Open source serves as an important pillar for most major successful commercial hardware and software businesses (e.g. Google, Facebook, Amazon, Android phone manufacturers, etc.), from the original academic research behind the electrical engineering and data structures, to many if not most of the programming APIs these commercialized technologies utilize. This process is driven by hundreds of mostly-anonymous contributors toiling away on their own use-cases, forking libraries and pushing updates as needed and generally in their spare time. Now people get hired specifically to improve open source libraries for companies that rely on them under the hood, which gets propagated to everyone else worldwide under open source licenses like the GPL license. Less strict open source licenses like LGPL, MIT (our license of choice for this project), or even the Unlicense allow monetization and privatization of offshoots of these libraries.

Every programming language in mainstream use is open source. Even Iphones are made of mostly open source, public university-developed technology. As transistor and miniaturization costs continue to plummet in the manufacturing world, it's now viable to even build your own Android phone. The Maker movement, centered around 3D printing and STEM education, relies entirely on open sourced patents in what was once a market restricted by only one company. Facebook company coders spend much of their resources on building powerful open source libraries, which fuels the adoption rate of their commercial products that are already integrated with said libraries. Open source tech is precisely the opposite of intellectual property, it's everyone's, while it leaves the door open for profit.

What this all reveals is one of the most important structures in the advancement of computer and human infrastructure. That's the open, honest collaboration between people just trying to get things done and make common utilities that much more efficient and smooth both for creators/engineers and consumers. This naturally lends to commercializing niches that require the specialized, time-consuming work to justify monetization - and help engineers and creators survive. This then informs the general functionalities that need to be further developed in the open source realm, which makes everybody's lives easier.

Many incredible commercial software platforms have been made free and open source because of this collaborative streamlining that happens when you have thousands of programmers investing their time for their own needs. Unreal Engine and Unity Engine, Sony's animation software, Google's data science APIs, etc. In spite of the millions of dollars required to develop these platforms, their progenitor companies were able to open them up for everybody to create more commercial grade products of their own. It was the information and resource asymmetry combined with their expert systems that let these groups produce these tools first, not any necessary specialness or exclusivity of the ideas which have become important for everyone like other kinds of infrastructure. This is causing massive gains in their own long term profits, especially as other creators begin to call these platforms home for their

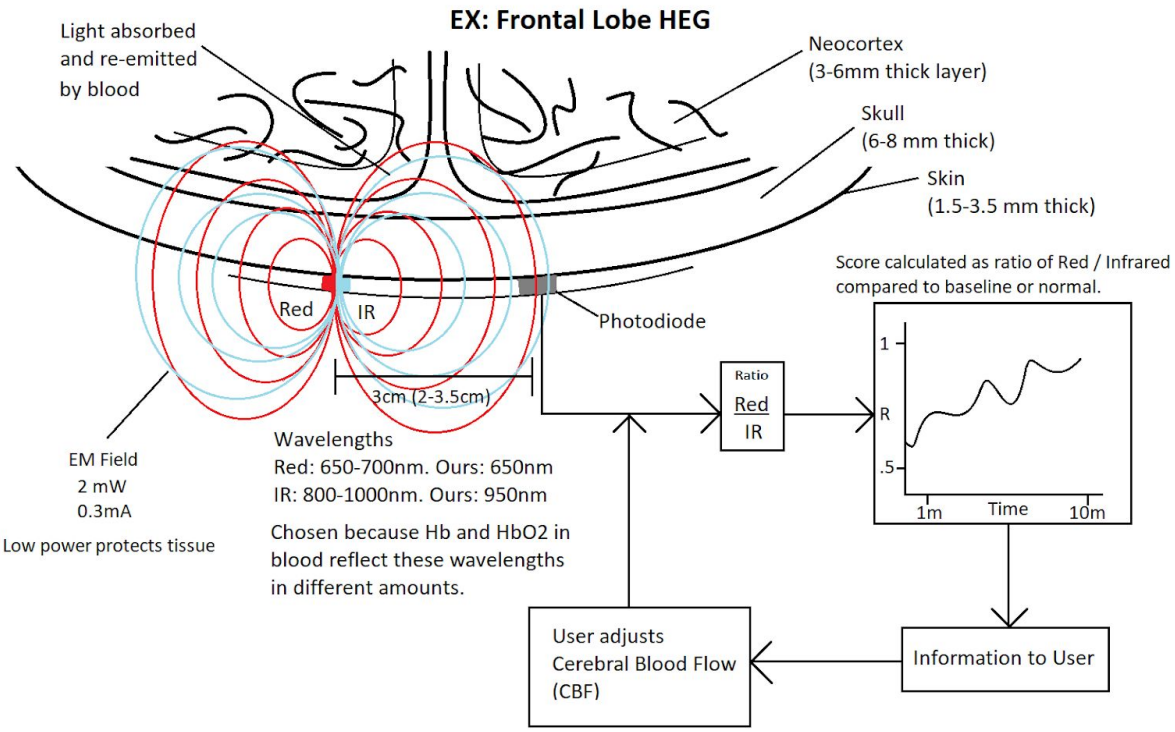
always up-to-date standards - made possible by the hundreds of additional collaborators in their own spaces who are all moving the computer field forward.

Above business and profit, open source provides endless educational material. The initial run of open source, arduino-based HEGs will be kits made in this vein, as it will maximize learning potential for creators and modders at any age. This is an important first step as the kits scale from a single unit being available and in play up to thousands without much difficulty and excellent cost scaling, and guarantees information proliferation that is most necessary for emerging tech or any fields of study and application in general. Good, ethical innovation is always the result of collective effort on several levels, even if it is not always immediately obvious. We believe this identifies the first and most important base that needs to be served, as these collective infrastructure-bound technologies truly belong to everyone.

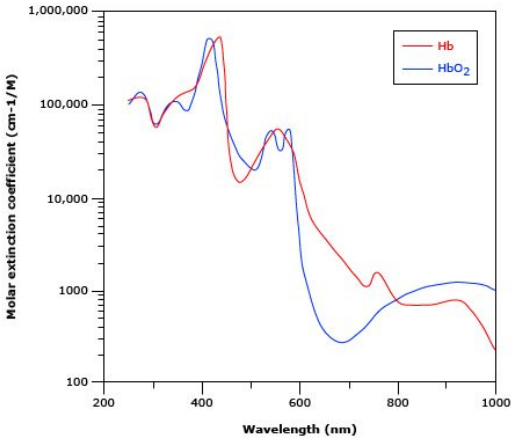
Open Source Hemoencephalography (HEG):

This HEG open source project is an attempt at creating a fundamental platform for others to build their own work on, and through the spirit of community roll whatever improvements they make to the general model into the open source resource pool. As the HEG itself is built entirely from maximally affordable open source tools, this leverages well-curated and deep learning resources with minimized risk to experimenters. There are ample opportunities to profit on top of this platform too, from building enhanced hardware and software packages for sale to running practices or doing research. A fully realized open source toolset will have a commercial quality on desktop and mobile and more right out of the box, so as to set a high quality standard for what content we believe should be free - which is most of it. The learning content will be academically up-to-date and to-standard with peer-reviewed sources, to set an intellectual and ethical bar that should only be surpassed by further contributions.

Theory:



Hemoencephalography is a very cheap, efficient method to gain deeper insight into the health of the outer regions of the brain - the neocortex and white matter circuits. [1] It does this by measuring blood perfusion and metabolic activity through the skull in whatever area targeted by the Red/Infrared sensor, and is very similar to basic pulse oximetry. With readings, users can measure [2] and gain control of their own brain blood flow and metabolic activity in targeted regions, to do what have been coined “brain push-ups,” or exercises meant to improve the quality of blood flow in the target area of the neocortex, thus improving associated behaviors and functions. [3,4] It takes 5 minutes in a single session to gain control, unlike other methods which take far longer with less reliability. This was originally developed and patented in the 90's to treat ADD as a purified, non-invasive, and safe alternative to generally less reliable and more expensive EEG, SPECT, MRI, etc. biofeedback methods [5] of measuring brain activity and giving useful physical information back to the user.

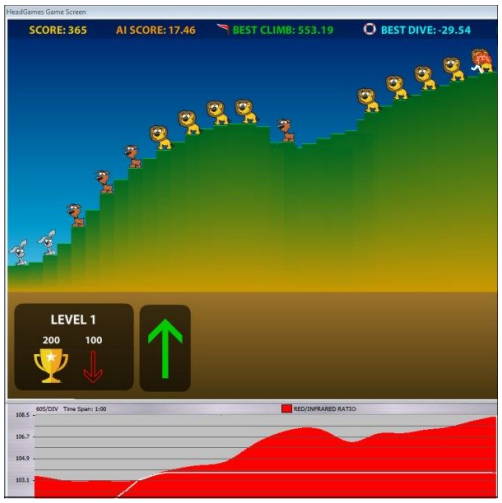


Left: Light Absorbance of Hb vs HbO2. *Right: Dense capillary networks cover the brain*

<1https://somepomed.org/articulos/contents/mobipreview.htm?13/55/14197> <http://www.sbirc.ed.ac.uk/cyril/MEEG4.html>

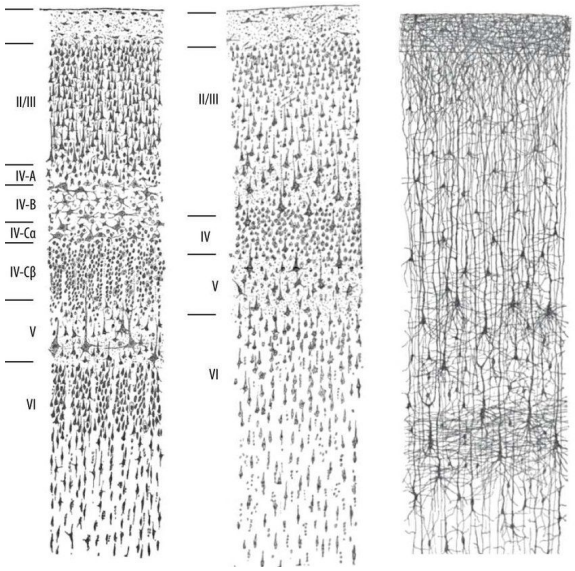
The neocortex is a layer 3-6mm thick that wraps around the outer part of your brain. It makes up about 76% of the brain's volume, uses about 80% of the blood and nutrient supply to the brain (30%-40% of the body's overall energy), and contains the most complex and mysterious natural circuitry we know of. It is unique to mammals, and humans have one of the largest brains relative to body size. [6]

The neocortex stores memories from all the senses, synthesizes language, makes up the frontal lobe that makes empathy and socialization possible, and even contains spatial maps of our physical world – as discovered in the winning experiment for the 2014 Nobel Prize in Medicine. It is broken up into many unique areas, from the empathy, logic, and language centers in the front and sides to the visual, auditory, and other sensory and movement processors toward the back. There can be a lot of crossover and overlap in the functions of these areas, something we owe our mental adaptability to as humans. In a similar vein, some symptoms are universal to mental health issues, like Hypofrontality, where the majority of the blood leaves our frontal lobe as part of our physical stress response, and can be mitigated with therapy. [7]



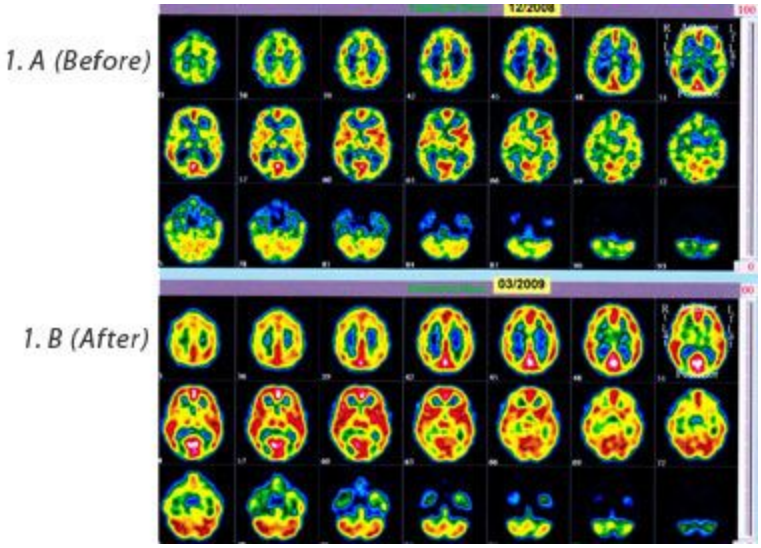
Simple HEG LIFE game by Brain Trainer

Knowing these facts, in tandem with current research on abnormal psychology like ADD, Depression (incl. Bipolar), Schizophrenia, PTSD, Dementia, and migraines to name a few... and in observation of emerging fields like Network Neuroscience and Behavioral Economics, it can be said with certainty that the general health of this neocortex - a dominant physiological feature that sets us apart from other animals – is an essential and often overlooked question when addressing human needs. It needs care, exercise, and stimulation like other parts of the body, and in more complex ways like the problems it developed to handle.



Adult vs toddler neocortex grey matter stains in various lobes.
<https://www.researchgate.net/figure/Drawing-of-neocortical-layers-from-Ramon-y-Cajal-1911-Left-Nissl-staining-of-the-adult_fig3_281183803>

HEG biofeedback is complementary to what most psychiatry and psychological treatments offer. It is cheap, safe, and easy enough to use to make it readily available to a much wider market than as purely a treatment tool – but also as a way to learn about and improve ourselves in general, especially when we have a very uninformed general population. We are at all-time high rates of mental illness and dependence in the United States, also 300 million people diagnosed with depression globally – recognized by the World Health Organization as one of the main contributors to disability and lost economic opportunity. [8]

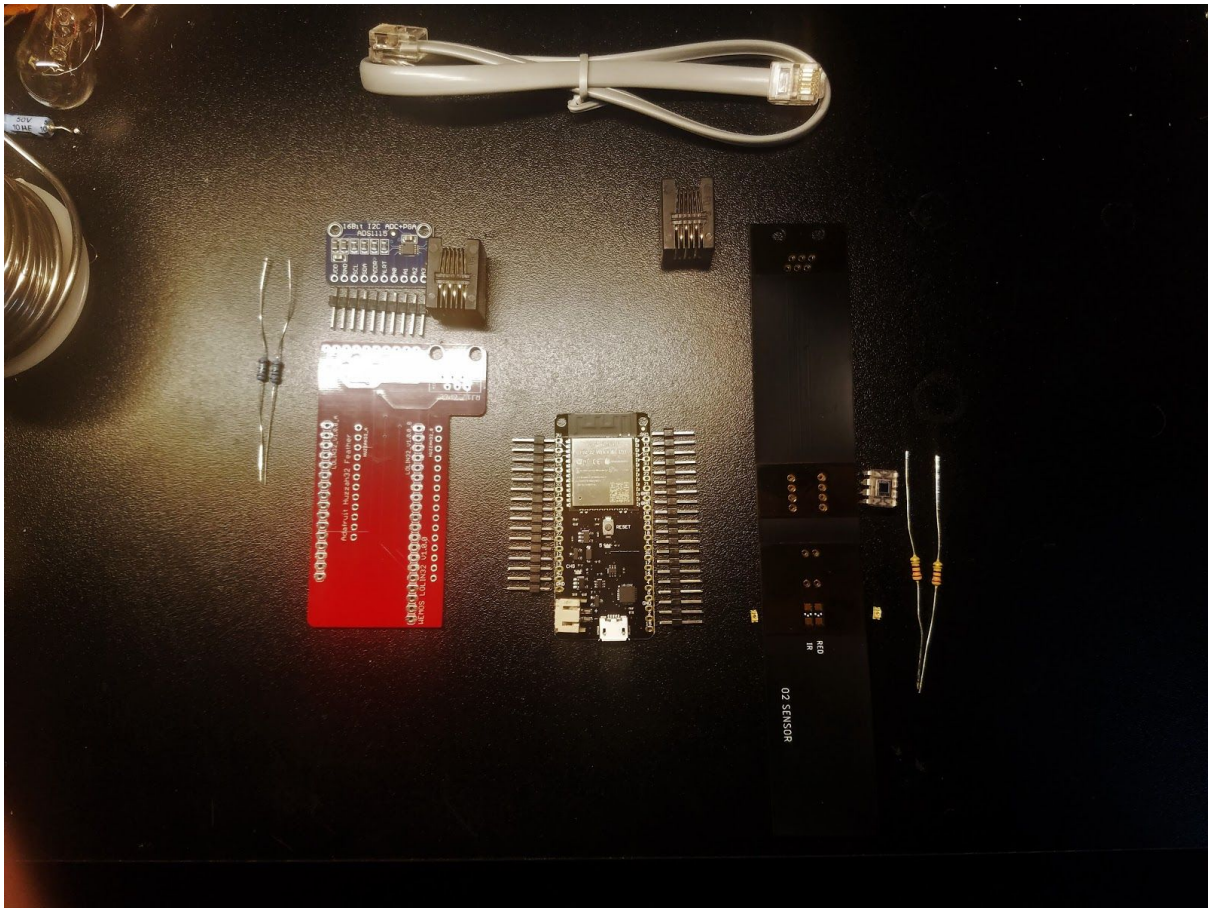


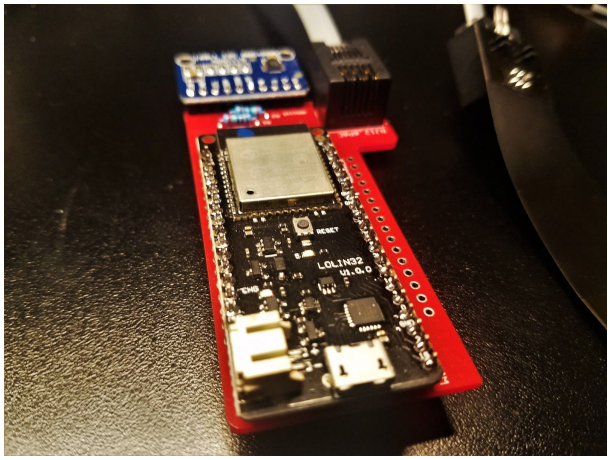
SPECT Image of a former drug abuser's brain pre- and post-HEG treatment.
<http://smtp.biocompresearch.org/biofeedback-institute-stories.html>

The "Archetypal HEG":

We are still working out minor details on the schematics for the final version, thus this whitepaper is subject to updates. However, full board files and schematics will be provided at the right time so as to make the open source HEG kits fully transparent and duplicable. There is immediate incentive to manufacture HEGs as they are not widely known or proliferated, while the information is mostly new and compelling (yet tied to long-standing health and wisdom traditions) - with everyone as a potential target user. Since we planned ahead, we will have sales of our archetypal "HEGduino" kit right as we are rolling out the open source launch. This grants us strategic advantage in that we should be able to stay ahead of the game as competitors begin to appear, while we have a fully realized product already under our belts. People tend to want to support the source, too, both because they simply want to be supportive or because of the authority that source contributors are granted for doing most of the work.

What is the archetype? This is the model to be the progenitor of further models. We designed a kit because it minimizes our manufacturing costs, the costs to buyers, and the risk of device failure when tinkering. We can incentivize engineers with free kits, free replacements, and co-collaborator name features (which look great on resumes). The circuitry itself is bare minimal, lacking several common features of other pulse oximeters, while still permitting the same amount of interactivity for biofeedback. The code is written in plain language with free programming tools. These were conscious choices for better cost scaling as well as demystifying the tech to tinkerers through the reduced complexity.



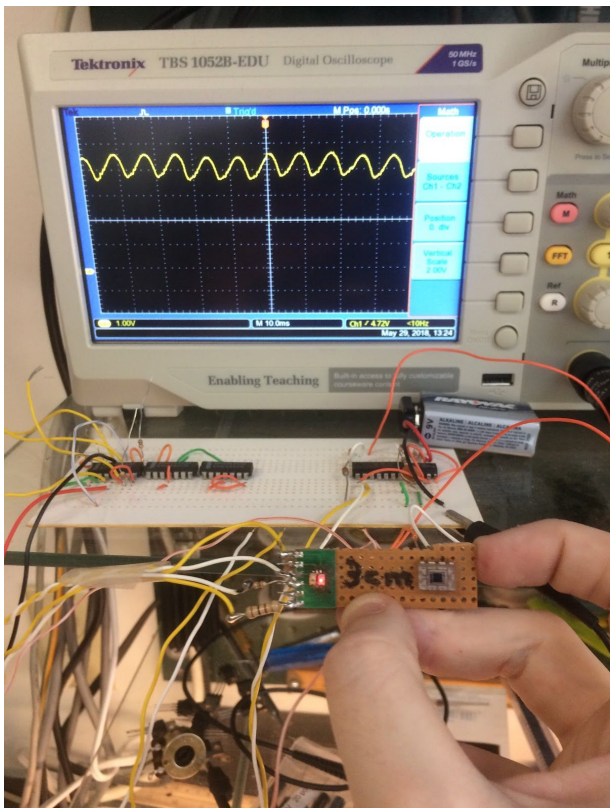


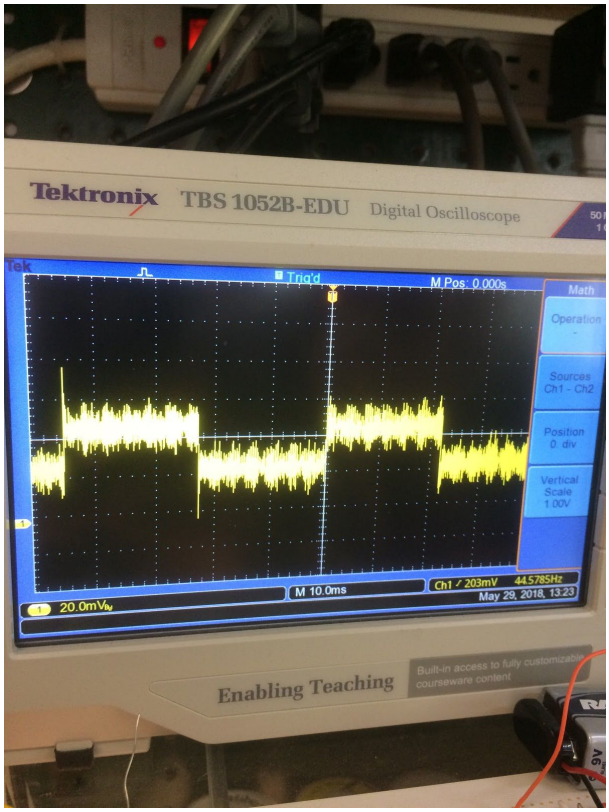
HEGduino kits, sponsored by [CrowdSupply](). Features a flex board with stiffener, easy mounting of generic arduino components, and printed instructions on the boards.

Test Setup and Tech Specs:

Test Setup:

Recreating patent 5995857 specifications:





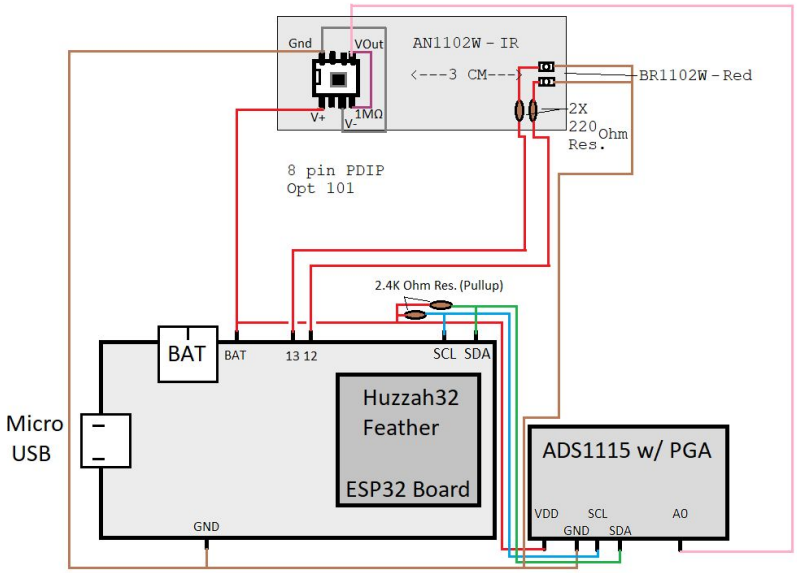
The original HEG sensor circuit is simply an LED flash timer and a buffer (the circuit on the right in the first image). The right image is the oscilloscope output result when placing the sensor on the forehead. You can see the signal is quite strong, within the mV range even with these low intensity LEDs. The noise is due to poor wiring and filtering. The higher amplitude square wave is the infrared pulse, the lower one is the red pulse. For comparison, the left image shows the results of the sensor under fluorescent light.

Sensor:
1x OPT101P Monolithic Photodiode
1x HAN1102W IR LED - 950nm @ 20mA*
1x BR1102W RED LED - 650nm @ 20mA

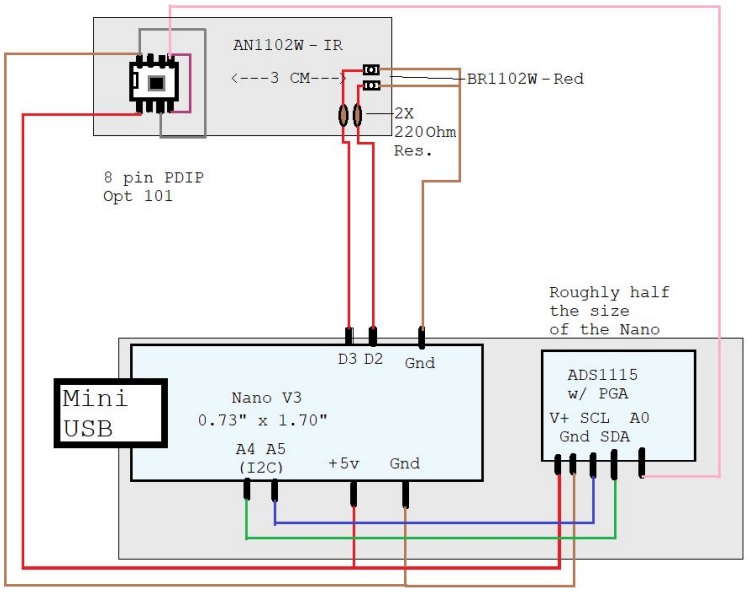
** DN1102W 850nm @ 20mA recommended by Biocomp*

Arduino Setups:

BT/Wifi-capable microcontroller:
ESP32 dev board with arduino driver, we used Adafruit Huzzah32 and the Wemos Lolin32 series:



USB-only microcontroller:
Arduino Nano v3 (ATmega328p-based) will suffice:



The sensor specifications follow the original patent 5995857 [1]. The lights are 20mA 940nm IR and 850nm Red LEDs, and the sensor is a generic OPT101 monolithic photodiode, common in pulse ox setups. The OPT101 is placed 3cm away from the LEDs to get an average

reading of LED light bouncing through the skull and brain surface. This spacing ensures that we are minimizing light reflected from the skin while not being too far to lose the light being bounced off the brain. The reading is then an average of returned light plus passive infrared (i.e. body heat), as long as all external light is blocked out from the sensor..

The analog-to-digital converter (ADC) is an ADS1115 16-bit ADC with a Programmable Gain Amplifier (PGA). The ADC functions at up to 860 samples-per-second (SPS) with 32767 steps, and a step resolution of about 0.125mV at max gain. The LED light from the sensor provides plenty of signal for the ADC's fairly rough resolution to detect on top of the body's passive IR/heat. This can be used across 4 channels. We can upgrade this to an ADS1256 for example, which has 24 bits available and a 30kSPS sample rate and 8 channels, more ideal for simultaneous multi-sensor setups and gathering clinical data.

The microcontroller simply requires an I2C input for these types of ADCs and preferably a serial output, whether via USB or wirelessly via 802.11g or Bluetooth. The most affordable microcontrollers out there, like the ATmega328P-based Nano v3 - found for less than 2 dollars a pop on Aliexpress, have enough horsepower to take all the readings necessary at our sample rates. The ESP32 can handle these readings plus a WiFi, Bluetooth (including BLE), or USB output. The Bluetooth response rate is restricted on the ESP32 in the arduino libraries to 10ms, but one can send arrays of amassed data in those periods to compensate for the bottleneck, or work around it with the standard C libraries. Arduino, or even Raspberry Pi, provide ideal starting points for newcomers in hardware and for setting a minimum bar for commercial production.

Code

Available on github.

Code breakdown of HEG_BTSerial.ino for ESP32 arduino boards as of 3/7/2019. This code and walkthrough are geared for novices. It won't cover every line but the code itself is well-commented and self-explanatory if you follow the logic. Using Visual Studio Code in images.

Required:

ESP32-based Arduino board. Recommended: Wemos or Adafruit brands, lots of options.
Arduino IDE
ADS1X15 library
Arduino_ESP32 library

Lines 23-30

```
23  bool USE_USB = true; // WRITE 'u' TO TOGGLE, CHANGE HERE TO SET DEFAULT ON POWERING THE DEVICE
24  bool USE_BLUETOOTH = true; // WRITE 'b' TO TOGGLE
25  bool pIR_MODE = false; // SET TO TRUE OR WRITE 'p' TO DO PASSIVE INFRARED ONLY (NO RED LIGHT FOR BLOOD-OXYGEN DETECTION). RA
26
27  bool DEBUG_ESP32 = false;
28  bool DEBUG_ADC = false; // FOR USE IN ARDUINO IDE WITH VIEW_ADC_VALUE
29  bool DEBUG_LEDS = false;
30  bool SEND_DUMMY_VALUE = false;
31
```

Set default settings whether you want to use USB or Bluetooth input/output, or both.concurrently. These can be toggled via command, too, but you need at least one active by default. The last setting is whether you want to enable pIR mode or not, which is experimental at the moment but requires no LEDs, as pIR is merely correlating passive body heat with metabolic activity.

The different Debug bools are helpful in the arduino IDE, to let you check out different values in the serial monitor and plotter.

One more notable value is SEND_DUMMY_VALUE. This will send random values in the same string format as the real data so you can simply test throughput. Not shown, setting sensorEnabled to true will have the sensor run upon powering the device by default rather than requiring toggling with ‘t’/’f’.

Lines 38-40

```
40  // PUT IR IN 13, RED IN 12
41  const int IR = 13;
42  const int RED = 12;
43  const int LED = 5; // Lolin32 V1.0.0 LED on Pin 5
44  int16_t adc0; // Resulting 15 bit integer.
45
```

Set pins for the RED and IR sensor LEDs. The third LED setting is if your ESP32 board has an indicator LED, as demonstrated on the Lolin32. Pins 13 and 12 are the chosen defaults for the Lolin32 and Huzzah32 setups.

Lines 97-101

```
96  //Make sure these divide without remainders for best results
97  const int ledRate = 50; // LED flash rate (ms). Can go as fast as 10ms for better heartrate visibility.
98  const int sampleRate = 1.5; // ADC read rate (ms). ADS1115 has a max of 860sps or 1/860 * 1000 ms or 1.16ms
99  const int samplesPerRatio = 5; // Minimum number of samples per LED to accumulate before making a measurement.
100 const int BTRate = 100; // Bluetooth notify rate (ms). Min rate should be 10ms, however it will hang up if the
101 const int USBRate = 0; // No need to delay USB unless on old setups.
102
```

These are the timing variables (in milliseconds) for the various timed events. ledRate sets the flash alternation rate. Currently this code is set to flash between Red and Infrared with no rest period where both LEDs are disabled, which is recommended to get a baseline adc reading that can be subtracted as noise from the LED signal. It is not necessary to see changes expressed, as ratio to blood-oxygen is on a nonlinear scale anyway and the ADC is plenty sensitive to pick them up on top of the body noise.

sampleRate is the rate at which the ADC takes readings. The ADS1115 max read rate is 860 samples-per-second or 1/860 seconds per tick. We set it to 1.5 milliseconds for about 667 samples per second, which is more than enough. Samples for each LED are averaged over the period of each LED flash. This averaging reduces noise and lets one take full advantage of the ADC.

samplesPerRatio is the minimum number of samples for each LED flash required to take a ratio. We recommend setting it based on a full flash cycle.

BTRate is the Bluetooth transmission rate. It's set to 100ms by default, can go to 10ms in the Arduino IDE, while it should be able to go much faster through the C libraries provided by Espressif, which are much harder to code with. 100ms is plenty for user interaction, while USB data transmission is always as fast as the sample rate. Bluetooth data is averaged before sent, in case 2 or more ADC or ratio readings are taken per Bluetooth interval. This allows variables to be changed arbitrarily while minimizing loss. USBRate default set to 0 as delay is not needed unless on slower computers.

Lines 104-117

```
103  //Start ADC and set gain. Starts timers
104  void startADS() {
105    // Begin ADS
106    ads.begin();
107    ads.setGain(GAIN_SIXTEEN);
108
109    //ads.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit = 3mV (default)
110    //ads.setGain(GAIN_ONE);      // 1x gain   +/- 4.096V  1 bit = 2mV
111    //ads.setGain(GAIN_TWO);      // 2x gain   +/- 2.048V  1 bit = 1mV
112    //ads.setGain(GAIN_FOUR);     // 4x gain   +/- 1.024V  1 bit = 0.5mV
113    //ads.setGain(GAIN_EIGHT);    // 8x gain   +/- 0.512V  1 bit = 0.25mV
114    //ads.setGain(GAIN_SIXTEEN); // 16x gain  +/- 0.256V  1 bit = 0.125mV
115
116    adcEnabled = true;
117  }
```

The startADS() function begins the ADC reading and sets the gain to the maximum. The program and LED timers start at this moment, too. For boards without pre-set SDA and SCL pins, you can use Wire.begin(SDA,SCL) instead of ads.begin().

Lines 119-162

commandESP32()

We are using simple 1 character commands over full strings. This is easier to deal with when using BLE instead of a serial profile. Commands are:

t = enable sensor and HEG program
f = disable sensor and HEG program
r = reset baseline and ratio readings
p = pIR mode. Disables LEDs basically
u = toggle USB output
b = toggle Bluetooth output
0-3 = Sending ‘0’ thru ‘3’ will change which of the 4 channels the ADS1115 is using.

Lines 164-178

```
164  void setup() {
165    if (USE_USB == true) {
166      Serial.begin(115200);
167    }
168    pinMode(IR, OUTPUT);
169    pinMode(RED, OUTPUT);
170    //LOLIN32 ONLY
171    pinMode(LED, OUTPUT);
172    digitalWrite(LED, HIGH);
173
174    //esp_bt_sleep_disable(); // Disables sleep mode (debugging)
175    SerialBT.begin("My_HEG");
176    BLEMillis = millis();
177    USBMillis = millis();
178  }
```

One of the default functions for arduino. This sets up GPIO pins and begins advertising the bluetooth server. The default name for Bluetooth pairing is “My_HEG”, which will show up upon scanning for devices on phone or laptop.

Lines 181-188

```
181  void core_program() {
182    if (sensorEnabled == true) {
183      if (adcEnabled == false) {
184        startADS();
185        //Start timers
186        sampleMillis = millis();
187        ledMillis = millis();
188      }
```

The core_program() contains the essential HEG code. It begins when the sensorEnabled bool is set to true. The program then checks if the adc is enabled and enables it, then begins reading the ADC. After the ADC is enabled, the LED flash sequence will begin, followed by sampling.

Lines 192-228

```
189     if (SEND_DUMMY_VALUE != true) {
190       // Switch LEDs back and forth.
191       // PUT IR IN 13, AND RED IN 12
192       if (currentMillis - ledMillis >= ledRate) {
193         if (red_led == true) { // IR on
194           if (pIR_MODE == false) {
195             digitalWrite(RED, LOW);
196             digitalWrite(IR, HIGH);
197             if(DEBUG_LEDS == true) {
198               Serial.println("IR ON");
199             }
200           }
201           red_led = false;
202           ir_led = true;
203           no_led = false;
204         }
205         else if ((red_led == false)){// && (no_led == true)) { // Red on
206           if (pIR_MODE == false) { // no LEDs in pIR mode, just raw IR from body heat emission.
207             digitalWrite(RED, HIGH);
208             digitalWrite(IR, LOW);
209             if(DEBUG_LEDS == true){
210               Serial.println("RED ON");
211             }
212           }
213           red_led = true;
214           ir_led = false;
215           no_led = false;
216         }
217         else { // No LEDs
218           digitalWrite(RED,LOW);
219           digitalWrite(IR,LOW);
220           if(DEBUG_LEDS == true){
221             Serial.println("NO LED");
222           }
223           red_led = false;
224           ir_led = false;
225           no_led = true;
226         }
227         ledMillis = currentMillis;
228       }
```

LED alternating sequence, contains commented out code to add a third no-LED sequence. Change ledRate to change the interval period for the LEDs, in milliseconds per LED

Lines 230-268

```
230     if (currentMillis - sampleMillis >= sampleRate) {
231       // read the analog in value:
232       adc0 = ads.readADC_SingleEnded(adcChannel); // -1 indicates something wrong with the ADC
233       //Voltage = (adc0 * bits2mv);
234 
235       // print the results to the Serial Monitor:
236       if (DEBUG_ADC == true) {
237         Serial.println("ADC Value: ");
238         Serial.println(adc0);
239         //Serial.println("\tVoltage: ");
240         //Serial.println(Voltage,7);
241       }
242       else {
243         if ((adc0 >= 3000) || (reset == true)) { // The gain is high but anything over 3000 is m
244           //Serial.println("\nBad Read ");
245           badSignal = true;
246 
247           //Temp: reset baseline on bad read
248           signalDetermined = false;
249           reset = false;
250           baseline = 0;
251 
252           ticks0 = 0; // Reset counter
253           ticks1 = 0;
254           ticks2 = 0;
255           ticks5 = 0;
256           redValue = 0; // Reset values
257           irValue = 0;
258           rawValue = 0;
259           redAvg = 0;
260           irAvg = 0;
261           rawAvg = 0;
262           ratioAvg = 0;
263           posAvg = 0;
264         }
265         else {
266           if (badSignal == true) {
267             badSignal = false;
268           }
```

If the sampleRate time has elapsed, the device reads the ADC on the I2C channel. If the adc signal is over 3000 or the reset flag has been triggered, the ratio scoring and baseline will be reset. This ensures that there is a rough margin for bad data. If the signal is under 3000, the badSignal flag will be set to false and the baseline will begin accumulating.

Lines 269-289

```
269             if (signalDetermined == false) { // GET BASELINE
270               ticks0++;
271               if (ticks0 > 250) { // Wait for 500 samples of good signal before getting baseline
272                 // IR IN 12, RED IN 13
273                 if ((ticks1 < 250) && (ticks2 < 250)){// && (ticks5 < 250)) { // Accumulate samples for baseline
274                   if (red_led == true) { // RED
275                     redValue += adc0;
276                     ticks1++;
277                   }
278                   else if (ir_led == true) { // IR
279                     irValue += adc0;
280                     ticks2++;
281                   }
282                   else {
283                     rawValue += adc0;
284                     ticks5++;
285                   }
286                   //Serial.println("\nGetting Baseline. . .");
287                 }
288                 else {
289                   signalDetermined = true;
```

This section is for determining a baseline value, which is used to compare a position change. After the badSignal flag is set to false, the device waits for 250 samples (to allow time for the sensor to settle) then accumulates 250 samples of each LED. There is a commented out extra option for comparing ratios after subtracting non-LED data out. After the sample quota is met, signalDetermined is set to true and the initial baseline ratio is calculated.

Lines 290-314

```
288                 else {
289                   signalDetermined = true;
290
291                   //rawAvg = rawValue / ticks5;
292                   redAvg = log10((redValue / ticks1));// - rawAvg);
293                   irAvg = log10((irValue / ticks2));// - rawAvg);
294
295                   baseline = (redAvg / irAvg) * 100; // Set baseline ratio, multiply by 100 for scaling.
296                   ratioAvg += baseline; // First ratio sent via serial will be baseline.
297                   bratioAvg += ratioAvg;
298
299                   redValue = 0; // Reset values
300                   irValue = 0;
301                   rawValue = 0;
302                   ticks0 = 0; // Reset counters
303                   ticks1 = 0;
304                   ticks2 = 0;
305                   ticks5 = 0;
306                   ticks3++;
307                   bticks3++;
308
309                   //Uncomment this for baseline printing
310                   //Serial.println("\tBaseline R: ");
311                   //Serial.print(baseline,4);
312                 }
313               }
314             }
315             else { // GET RATIO
```

Ratio is taken by dividing the logarithm of the Red light average (accumulated red samples / number of sample ticks) by the logarithm of the IR light average. The log scaling is not necessary but was recommended in a pulse oximetry paper [15]. We observed it slightly improved emphasis in ratio changes but reduces the range of variance. This might be removed later. The bratioAvg value, badc0, and bposAvg values are separate for bluetooth data, as it uses averaging to minimize loss over a slower transmitter interval.

Lines 315-340

```
315             else { // GET RATIO
316               ticks0++;
317               if (red_led == true) { // RED
318                 redValue += adc0;
319                 ticks1++;
320               }
321               else if (ir_led == true) { // IR
322                 irValue += adc0;
323                 ticks2++;
324               }
325               else {
326                 rawValue += adc0;
327                 ticks5++;
328               }
329               if ((ticks1 >= samplesPerRatio) && (ticks2 >= samplesPerRatio)){// && (ticks5 >= samplesPerRatio)) { // A
330                 //rawAvg = rawValue / ticks5;
331                 redAvg = log10((redValue / ticks1));// - rawAvg); // Divide value by number of samples accumulated // S
332                 irAvg = log10((irValue / ticks2));// - rawAvg);
333                 ratio = (redAvg / irAvg) * 100; // Get ratio, multiply by 100 for scaling.
334                 ratioAvg += ratio;
335                 bratioAvg += ratio;
336
337                 p1 = p2;
338                 p2 = ratio - baseline; // Position
339                 posAvg += p2 - p1;
340                 bposAvg += p2;
```

After baseline is achieved, the regular ratio-taking interval kicks in. The minimum samplesPerRatio should be however many samples happen over a single LED flash. The ADS1115 can get more than 20 samples per flash, although we are functioning closer to 6 samples per flash, and our default minimum is set to 5 samples required. This is ample for scoring at a 20Hz LED rate. posAvg tracks the slope of the ratio changes from baseline, providing a simple way to track changes over time. This can be used to calculate “velocity” and “acceleration”derivative curves for additional metrics. posAvg / time between p1 and p2 = “velocity.” “velocity” / time between p1 and p2 = “acceleration.”

There is more commented-out code below this screenshot that is not shown, useful for debugging.

Lines 370-394

```
370                     ticks0 = 0; //Reset Counters
371                     ticks1 = 0;
372                     ticks2 = 0;
373                     ticks5 = 0;
374
375                     ticks3++;
376                     bticks3++;
377
378                     redValue = 0; //Reset values to get next average
379                     irValue = 0;
380                     rawValue = 0;
381                   }
382                 }
383               }
384             }
385             sampleMillis = currentMillis;
386           }
387
388           adcAvg += adc0;
389           ticks4++;
390
391           badcAvg += adc0;
392           bticks4++;
393         }
394       }
395
```

After a ratio is taken, the LED sampling ticks and values reset, the sampling timer is updated, and the adcAvg with paired sampling ticks value accumulates the next reading.

Lines 410-479

The bluetooth() function controls information being sent via the Bluetooth-Serial port. It triggers every time the BLE interval passes. There is a bottleneck of 10ms for the bluetooth signal in arduino. The string format sent is “ADC,RATIO,POS”. While baseline and intermediate LED samples are being accumulated, the RATIO will read “WAIT” and no POS value will be reported. If SEND_DUMMY_VALUE is true the dummy string is generated on line 542. This lets you test computer or phone-side bandwidth easily without needing to assemble the HEG.

Lines 482-556

The usbSerial() function is essentially the same as the bluetooth() function minus the 10ms bottleneck.

Lines 558-573

checkInput() checks if any single char commands have been received over serial or bluetooth.

Lines 575-588

loop() is the default core loop for arduino scripts. On the ESP32 it automatically distributes work between cores. The core program, input checking, and usb and bluetooth transmission are handled here. The 1ms delay is placeholder as not having it blocks input checking.

HEGstudio modification

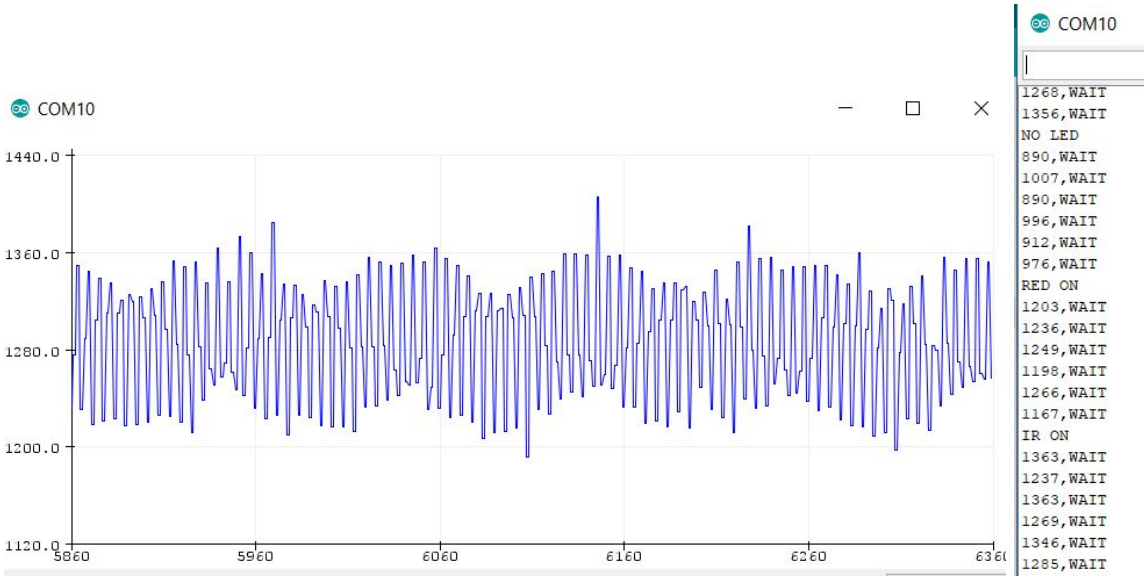
Our HEGstudio scoring code via modded exe or parsers.py in the source, available on github:

```
1136    def OnHEGduinoData(self, datum):
1137        print "RAW: ", datum
1138        self.heg_raw.append(datum) # currently receiving ratio. See SerialManager.run() and ProtocolDetector.run()
1139        length = len(self.heg_raw)
1140
1141        dy = 0 # Change in Amplitude from Baseline
1142        sma = 0  # Simple Moving Average
1143        score = 0 # Scoring = accumulated change in amplitudes from baseline
1144        if(length > 20):
1145            for data in itertools.islice(self.heg_raw, length - 21, length - 1):
1146                sma += data
1147            sma = sma / 20
1148            self.heg_sma.append(sma)
1149            print "SMA: ", sma
1150            if self.use_sma:
1151                self.hegdata.append(sma)
1152            elif self.use_scoring:
1153                if not self.hegdata:
1154                    score = sma # First score is baseline.
1155                else:
1156                    dataLength = len(self.hegdata)
1157                    if sma < self.heg_sma[dataLength - 1] + 0.0001 and sma > self.heg_sma[dataLength-1] - 0.0001:
1158                        score = self.hegdata[dataLength - 1]
1159                    else:
1160                        fastSMA = 0
1161                        for data in itertools.islice(self.heg_raw, length - 6, length - 1):
1162                            fastSMA += data
1163                        fastSMA = fastSMA / 5
1164
1165                        dy = fastSMA - self.heg_sma[dataLength - 1]
1166                        score = self.hegdata[dataLength - 1] + dy * 10 # smoke and mirrors
1167
1168                self.hegdata.append(score)
1169            else:
1170                self.hegdata.append(datum)
1171        return
```

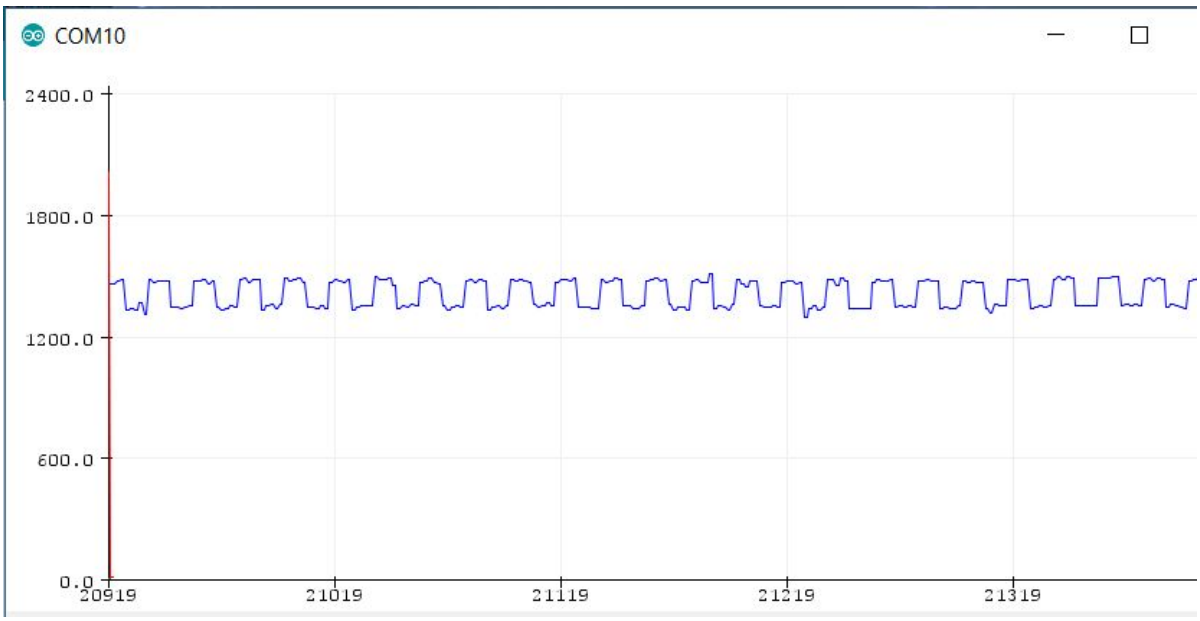
The scoring on HEG studio for our device is more of an “effort” score than an objective ratio, though it follows the ratio changes accurately. The scoring is simply a shorter-interval simple moving average minus a longer-interval simple moving average for the incoming ratio readings, added to the previous score, with some extra padding for noise and a multiplier to exaggerate changes. Raw light ratio changes are within the 1% to 5% range before applying the log10 scaling, which correlates to a full range of blood oxygen changes. We need more testing to determine the exact ratio to SpO2 relationship on our setup. This provides smoothing while still being reactive to changes as a sort of “effort” value. We don’t know if the other HEG solutions are close to this one, but this is good enough for real-time interactivity while not giving us an objective blood-oxygen ratio scale.

Test Results

Left: ADC results from sensor on forehead with no LEDS.
Right: ADC output showing LED reading differences with sensor on forehead.

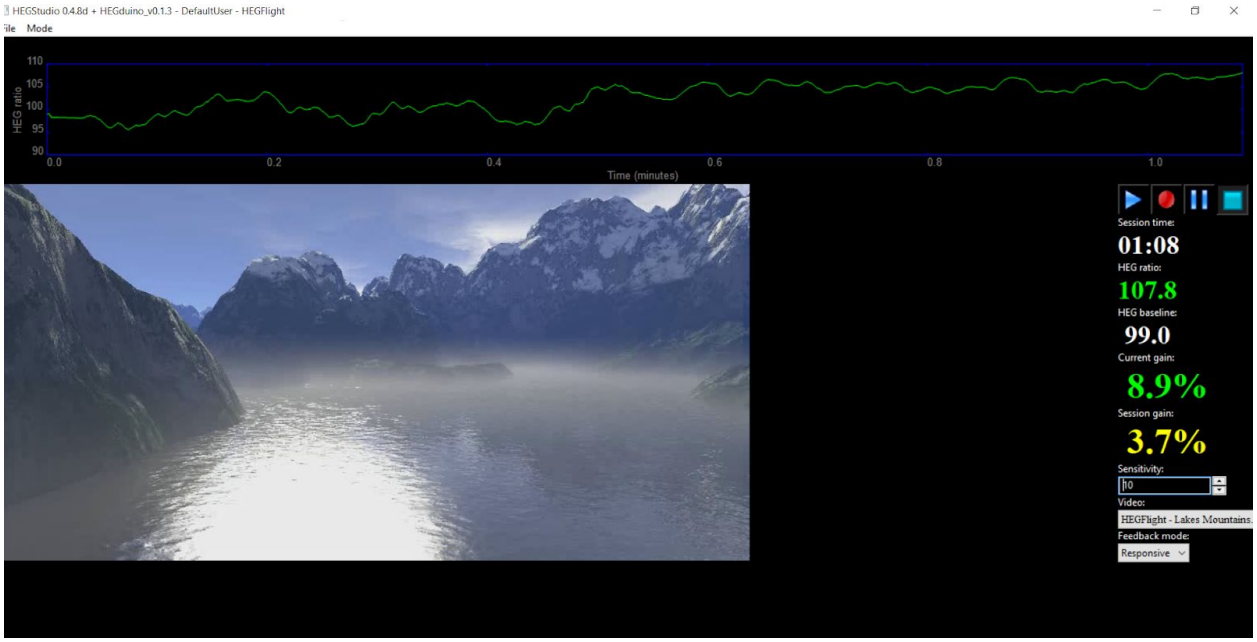


ADC output on Arduino IDE with LEDs alternating, sensor on forehead with no light leaking:



IR = higher values, Red = lower values

HEGstudio output using our scoring method:



Tests with about a dozen different users have shown this setup to successfully correlate with user effort. The score increases as the bloodflow ratio moving average increases or decreases. The reported current ratio here is not really the real ratio, as variance by our method is within the +/- 5% range without using a logarithmic scaling, and in the +/- 0.005% range after applying log10 scaling. Our SMA scoring function exaggerates changes and smooths feedback to be more user friendly and responsive.

Ideal Future Directions

Research

- Establish with data the clinical effectiveness of the HEG for various symptom profiles.
    - Also establish standard methodologies and interactivity to better address different symptoms.
    - Address ADD [9], Depression [10], PTSD [11], perhaps hallucinations [12], via stimulating and strengthening associated areas of the neocortex.
    - Exploring the effects of different types of stimulus on brain-blood flow quality/consistency, like mnemonics, music, games, etc. [13]
- Bring it into more schools and competitive environments.
    - Requires careful controls, software is the easiest way to provide these controls (as in providing predictable experiences with different objective outcomes).
    - Competitive gamers are an untapped market. Easiest to study, too.
- Geographic data from people walking around with this data, build stress and attention maps from these.
- Get deeper with the operant/classical condition capabilities - but don't be creepy or evil.
- MORE. DATA. QUANTIFY ALL THE THINGS.

Software

- Desktop Software
    - HEGstudio has been modded for compatibility with the HEG arduino profiles we provided at https://github.com/moothyknight/HEG_ESP32.
- Mobile Software
    - React Native and Nativescript attempts exist right now with performance issues.
- Unreal Engine and Unity Implementation, and more game engine and game content.
    - Will open up rapid prototyping UI and visual/sensory design greatly plus easy cross-platform.
    - Tap the competitive gaming market to provide game-integrated stress and focus monitoring or training cues.
    - Easy data science visualization.
    - Explore brain-computer interfacing and what types of feedback work.

Hardware

- Miniaturize. Cheapen. Miniaturize. Cheapen.
- Keep the open source model up to date with the popular hardware.
- Explore 2D and 3D mapping solutions.
    - 3D is the hardest but will create one of the first visible light body scanner type devices. Consider this Star Trek level.

Potential markets:

- 20% of the US population with some kind of mental illness, often go years without improvement or mere recognition of symptoms. [14]
- Students and children as a learning tool.
- Self-improvement and bio-hacking enthusiasts.
- Therapists as part of their toolkit (with approval as medical technology)
- All adds up to tens if not hundreds of millions of users.

Conclusion

We want this device to proliferate and have a major impact on how well people can treat their health and get to know themselves. We believe this would make the world a better place and open many more doors for mental health and biofeedback research, as HEG is currently too obscure and specialized for how straightforward and promising the science is. It's also convenient for people who do not have much time or energy to spare to their health, something key to consider in a country where people cannot or will not pay for co-pays or rehab, and general trust of professionals is low due to the dark and inexcusable history of mental health education and treatment in this country.

We'd like to extend special thanks to Eric Nichols, William Croft, Stephanie DuPont, Bob Marsh, Jonathan Toomim, and Peter van Deusen for their support and for the trailblazing they've done before this project that made it all possible

Citations/Further Reading:

1. Toomim, H, Marsh, R. "*Biofeedback of human central nervous system activity using radiation detection*" 1996. USPTO <https://patents.google.com/patent/US5995857A/en>
2. Bentourkia et al. "*Comparison of regional cerebral blood flow and glucose metabolism in the normal brain: effect of aging.*" 2000. PubMed. <https://www.ncbi.nlm.nih.gov/pubmed/11099707>
3. Dias et al. "*Clinical efficacy of a new automated hemoencephalographic neurofeedback protocol*" 2012. NCBI PubMed. <https://www.ncbi.nlm.nih.gov/pubmed/23156903>
4. Gomes et al. "*Hemoencephalography self-regulation training and its impact on cognition: A study with schizophrenia and healthy participants*" 2017. NCBI PubMed. <https://www.ncbi.nlm.nih.gov/pubmed/28882685>
5. Marzbani et al. "*Neurofeedback: A Comprehensive Review on System Design and Clinical Applications*" 2016. NCBI PubMed. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4892319/>
6. Rakic, Pasko. "*Evolution of the neocortex: Perspective from developmental biology*" 2009. NCBI Pubmed. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2913577/>
7. Galynker et al. 1998. "*Hypofrontality and negative symptoms in major depressive disorder.*" NCBI PubMed <https://www.ncbi.nlm.nih.gov/pubmed/9544664>
8. United Nations. "UN health agency reports depression now 'leading cause of disability worldwide'" Feb 2017. <https://news.un.org/en/story/2017/02/552062-un-health-agency-reports-depression-now-leading-cause-disability-worldwide#.WLRpQBB3xBw>
9. Sharma, Anup et al. "*Non-Pharmacological Treatments for ADHD in Youth*" 2016. NCBI PubMed. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4968082/>
10. PA. "*Schizophrenics learn to silence the voices thanks to MRI scanner, game.*" Feb 2018. The Sydney Morning Herald. <https://www.smh.com.au/world/europe/schizophrenics-learn-to-silence-the-voices-thanks-to-mri-scanner-game-20180212-p4z00t.html>
11. Toomim, H., Johnson, R. "*Anxiety, Anger, Depression, TBI and HEG*" Nov 2009. Futurehealth.org <https://www.futurehealth.org/articles/Anxiety-Anger-Depression-by-Hershel-Toomim-090723-101.html>
12. S, Yun et al. "*Stimulation of entorhinal cortex-dentate gyrus circuitry is antidepressive*" April 2018. NCBI PubMed. <https://www.ncbi.nlm.nih.gov/pubmed/29662202>
13. Dresier M et al. "*Mnemonic Training Reshapes Brain Networks to Support Superior Memory*" Mar 2017. NCBI PubMed. <https://www.ncbi.nlm.nih.gov/pubmed/28279356>
14. "Mental Health By The Numbers" 2018. NAMI.org. <https://www.nami.org/learn-more/mental-health-by-the-numbers>
15. Lopez, S. "*Pulse Oximeter Fundamentals and Design"* Nov 2012. Freescale Semiconductor. <https://www.nxp.com/docs/en/application-note/AN4327.pdf>

Further Reading:

16. Toomim et al. "*Intentional Increase of Cerebral Blood Oxygenation Using Hemoencephalography (HEG): An Efficient Brain Exercise Therapy*" 2005. Journal of Neurotherapy Vol 8 Issue 3. <https://www.tandfonline.com/doi/abs/10.1300/J184v08n03_02>
17. Shoshev, M. "*Hemoencephalography Neurofeedback - Mechanism of Action*" 2017. Trakia Journal of Sciences

<https://www.researchgate.net/publication/323699457_Hemoencephalography_neurofeedback_-_mechanism_of_action>

18. Zivodar, I. et al. "*Neurofeedback application in autistic spectrum disorders*" Sept 2015. ResearchGate. <https://www.researchgate.net/publication/290290548_Neurofeedback_application_in_the_treatment_of_autistic_spectrum_disorders_ASD>
19. Gagnon et al. "*Improved recovery of the hemodynamic response in diffuse optical imaging using short optode separations and state-space modeling*" 2011. NeuroImage Issue 56.
20. Gagnon et al. "*Short separation channel location impacts the performance of short channel regression in NIRS*" 2012. NeuroImage Issue 59.
21. Saager, R., Berger, A. "*Direct characterization and removal of interfering absorption trends in two-layer turbid media*" Sept 2005. Journal of Optical Society of America Vol 22.
22. Saager, R., Berger, A. "*Measurement of layer-like hemodynamic trends in scalp and cortex: implications for physiological baseline suppression in functional near-infrared spectroscopy*" 2008. Journal of Biomedical Optics Issue 13.
23. "*Functional near-infrared spectroscopy neuroimaging device to enter scene this year*" March 2016. Korea Advanced Institute of Science and Technology.
24. Marengo Rodriguez, F. "*Design and implementation of a low-cost near infrared spectroscopy system*" 2017. Self-published.
25. Muehlemann et al. "*Wireless miniaturized in-vivo near infrared imaging*" 2008. Optical Society of America.