Informe 6 Mentalink

Objetivos

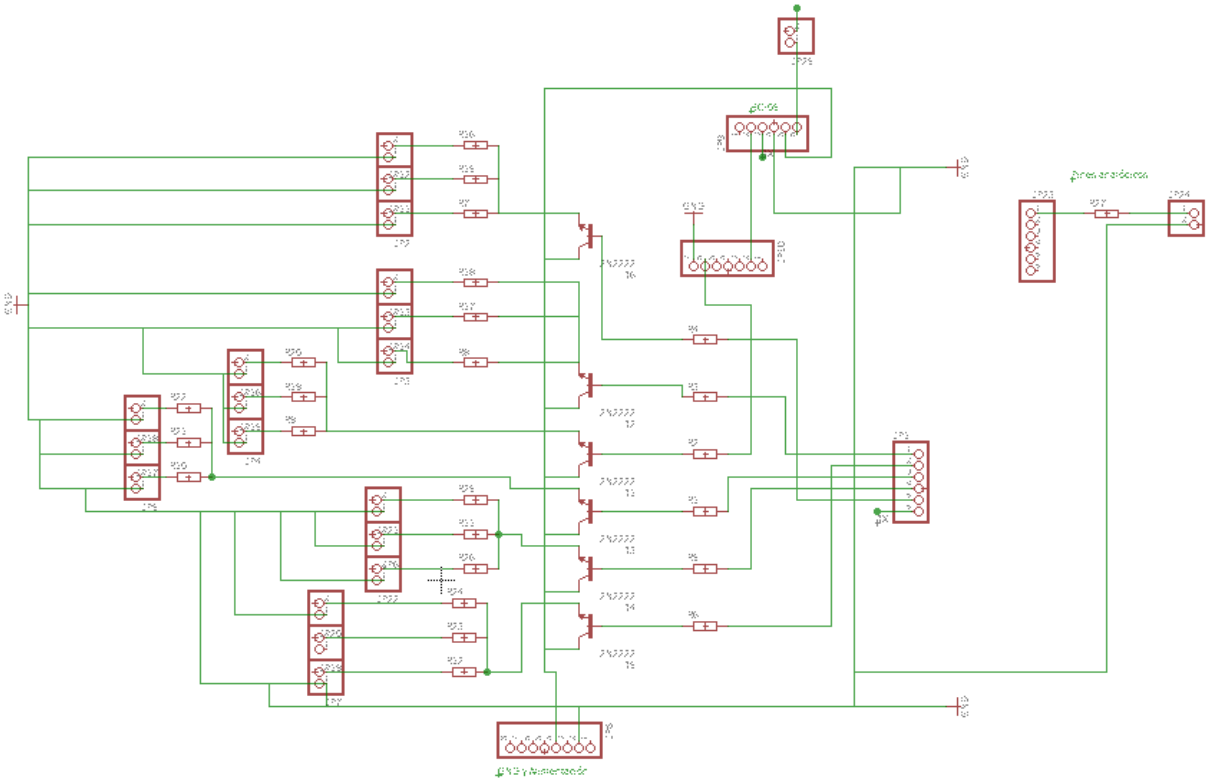
Contar que hemos hecho en este mes desde las presentaciones públicas del equipo.

Estimulador

En este momento realizamos el diseño e impresión para una consola con 6 estímulos, ya que queremos darle 6 órdenes al vehículo más 1 que sería el estado base detenido, pero este último no necesita de un estímulo ya que será la NO clasificación de ninguno de los 6.

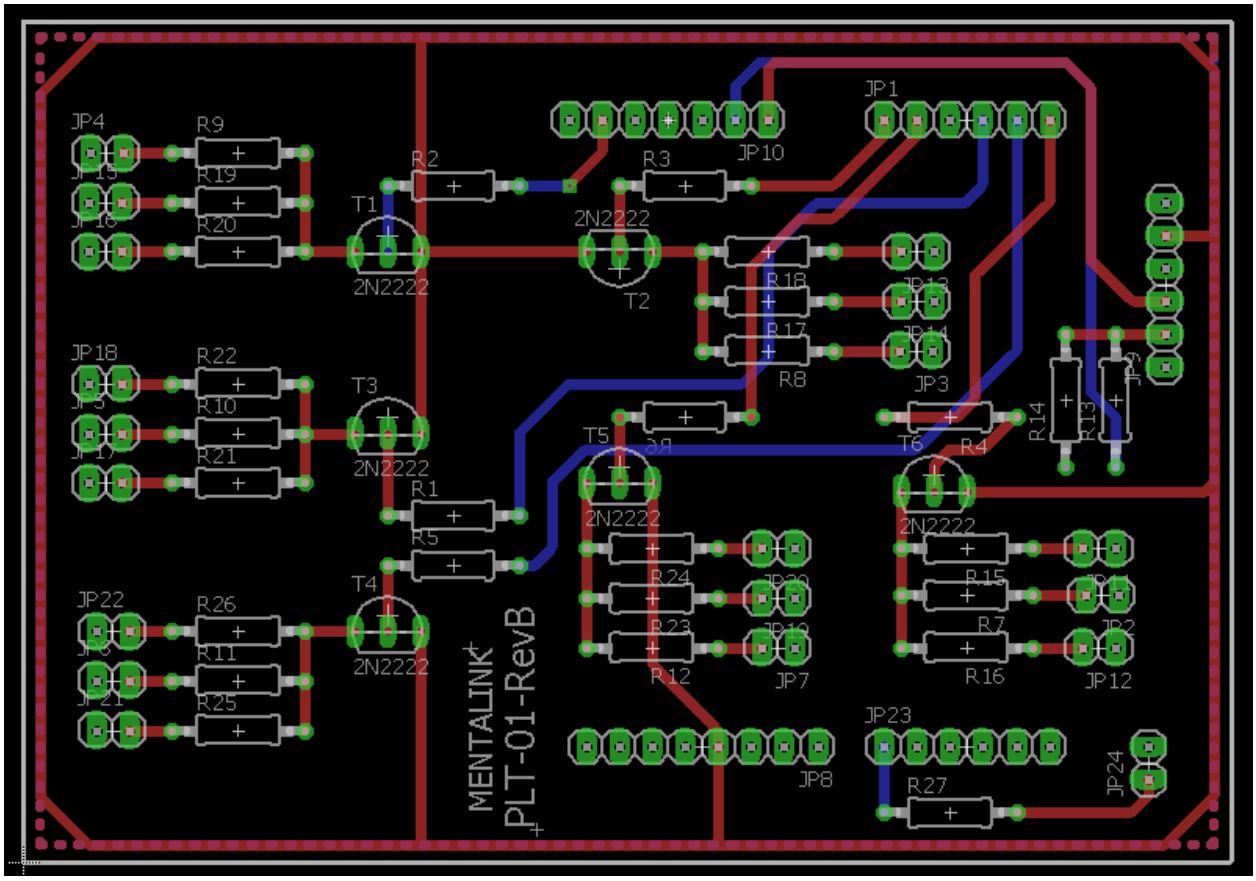
Esquemático

Luego de varias revisiones la última fue la C (la tercera), ya que la B tenía forma hexagonal y consumiría mucho material. Con ayuda del profesor Lucas quedó este esquemático:

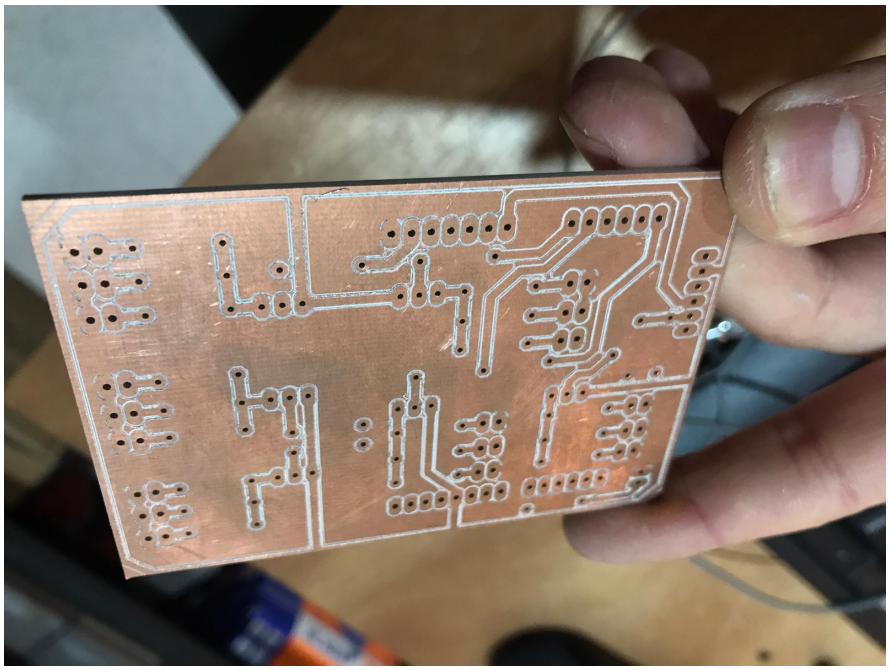


PCB

El respectivo PCB del diagrama esquemático anterior quedó así:



La impresión final quedó:



Más compacta y con ayuda de los PinHead irá sobre el Arduino Uno del Módulo 1, que es el que se encarga de controlar los estímulos y establece la comunicación BT.

Diseño de la consola

Clasificador

El objetivo del clasificador es: Clasificar la onda eeg.

Entendiendo clasificar como, determinar la frecuencia de mayor amplitud, que es la frecuencia del led observado por el sujeto. Dentro del proyecto la salida del clasificador se traduce a una instrucción y se envía al vehículo.

Para cumplir el objetivo planteamos 2 diferentes métodos por parte de nuestro equipo que estarán a prueba más adelante para definir cual usaremos (teniendo también en cuenta los otros clasificadores dados por la directiva). Esto dependerá de su precisión y velocidad clasificando eeg´s . La clasificación en estos dos métodos propuestos se hará a través de la transformada rápida de Fourier (*Detección de máximos en el espectro de la señal*) o una red neuronal(*RNN o Feed-Forward*).

Implementación por Red Neuronal
Creditos a DL4J API

Implementación con Feed Forward:

```
MultiLayerConfiguration discConf = new NeuralNetConfiguration.Builder()
        .seed(42)
        .updater(UPDATER)
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.IDENTITY)
        .gradientNormalization(GradientNormalization.RenormalizeL2PerLayer)
        .gradientNormalizationThreshold(100)
        .list()
        .layer(new DenseLayer.Builder().nIn(1114).nOut(2048).updater(UPDATER).build())
        .layer(new ActivationLayer.Builder(new ActivationReLU()).build())
        .layer(new DenseLayer.Builder().nIn(2048).nOut(1024).updater(UPDATER).build())
        .layer(new ActivationLayer.Builder(new ActivationReLU()).build())
        .layer(new DropoutLayer.Builder(1 - 0.5).build())
        .layer(new DenseLayer.Builder().nIn(1024).nOut(512).updater(UPDATER).build())
        .layer(new ActivationLayer.Builder(new ActivationLReLU(0.2)).build())
        .layer(new DropoutLayer.Builder(1 - 0.5).build())
        .layer(new DenseLayer.Builder().nIn(512).nOut(256).updater(UPDATER).build())
        .layer(new ActivationLayer.Builder(new ActivationLReLU(0.2)).build())
        .layer(new DropoutLayer.Builder(1 - 0.5).build())
        .layer(new OutputLayer.Builder(LossFunctions.LossFunction.XENT).nIn(256).nOut(12)
                .activation(Activation.SIGMOID).updater(UPDATER).build())
        .build();
```

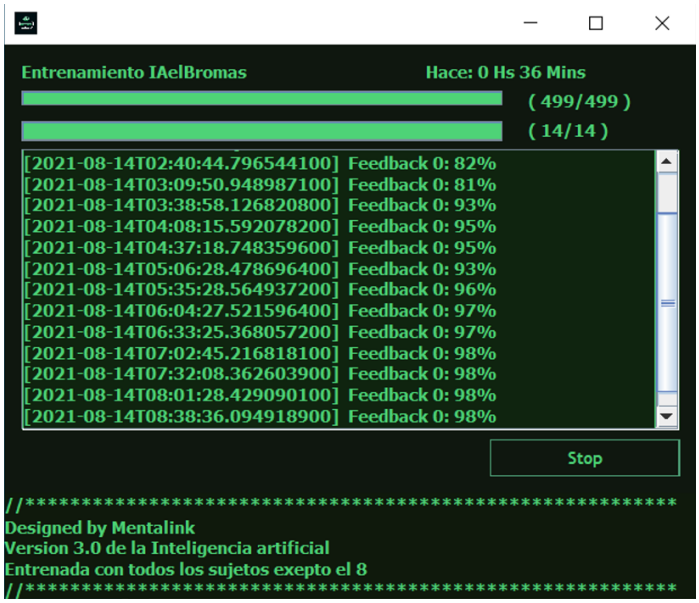
podríamos describir la inteligencia como un embudo que comprime y clasifica las entradas de 1114 a tan solo 12 (12 targets de frecuencia)

Su arquitectura es la siguiente

1. Una Input Layer
2. Fully Conected Layer (ReLu) que recibe los datos de la Input layer
3. Fully Conected Layer (ReLu) que baja la salida de 2048 a 1024
4. Fully Conected Layer (ReLu) que baja la salida de 1024 a 512

5. Fully Conected Layer (ReLu) que baja la salida de 512 a 255
6. Output Layer (Sigmoid y Xent) para obtener finalmente las 12 clases (12 targets)

Para entrenarla utilizamos todos los datos que se nos dieron en los datasets y creamos una interfaz la cual retorna la hora y la precisión que tiene la inteligencia en ese momento



El “hace 0 36 mins” está bugeado,
no sabía cómo programarlo y quedo mal.

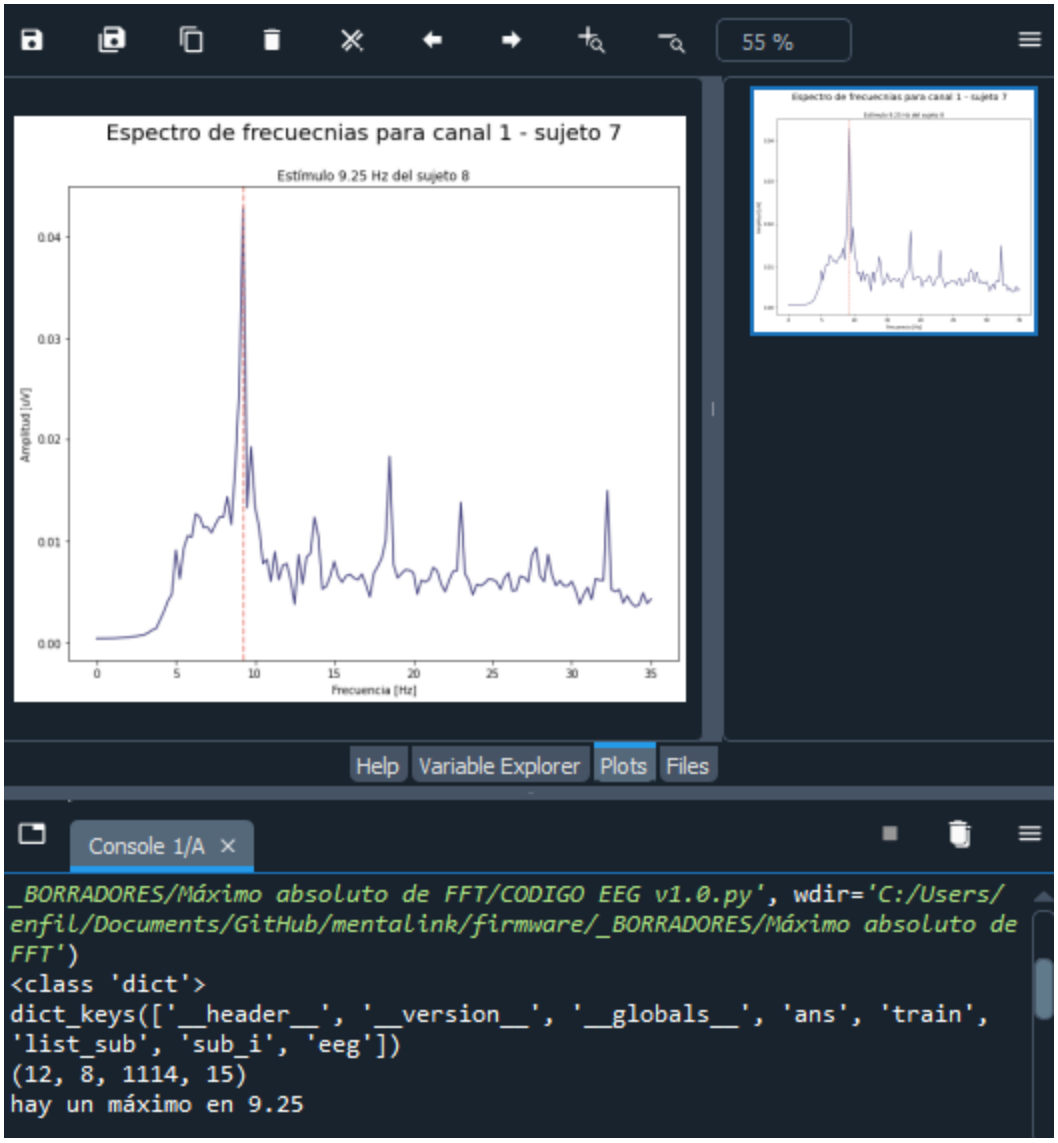
En la imagen no se ve pero inicialmente obtuvimos valores de 0,40 (40% precisión) en el inició del experimento (12:00 am) y con el tiempo el valor creció a 0,98 (98% precisión) (8:38 am)

Implementación por RNN

Dentro de las inteligencias artificiales se utiliza las recurrent neural networks para valores los cuales tienen una relación temporal entre ellos como lo puede ser Audio o una señal eeg
Lo más probable es que cuando la nueva inteligencia esté creada use LTSM layers con relu activation y sdg from classification y un output de softmax con

Detección de máximos en el espectro de la señal

Retomamos la clasificación mediante el método de máximos en el espectro de la señal, pero ahora lo que buscamos es el máximo absoluto de todo el espectro de frecuencias y no uno que se encuentre sobre el umbral. Ya que el método de umbrales es idealmente utilizado cuando se necesita un máximo pero en ese preciso momento, como la detección del END TIDAL de capnografía o para controlar las pulsaciones en un ECG. En nuestro caso el vector de datos ya está guardado en una lista y por lo tanto solo debemos recorrerlo y buscar su máximo absoluto.



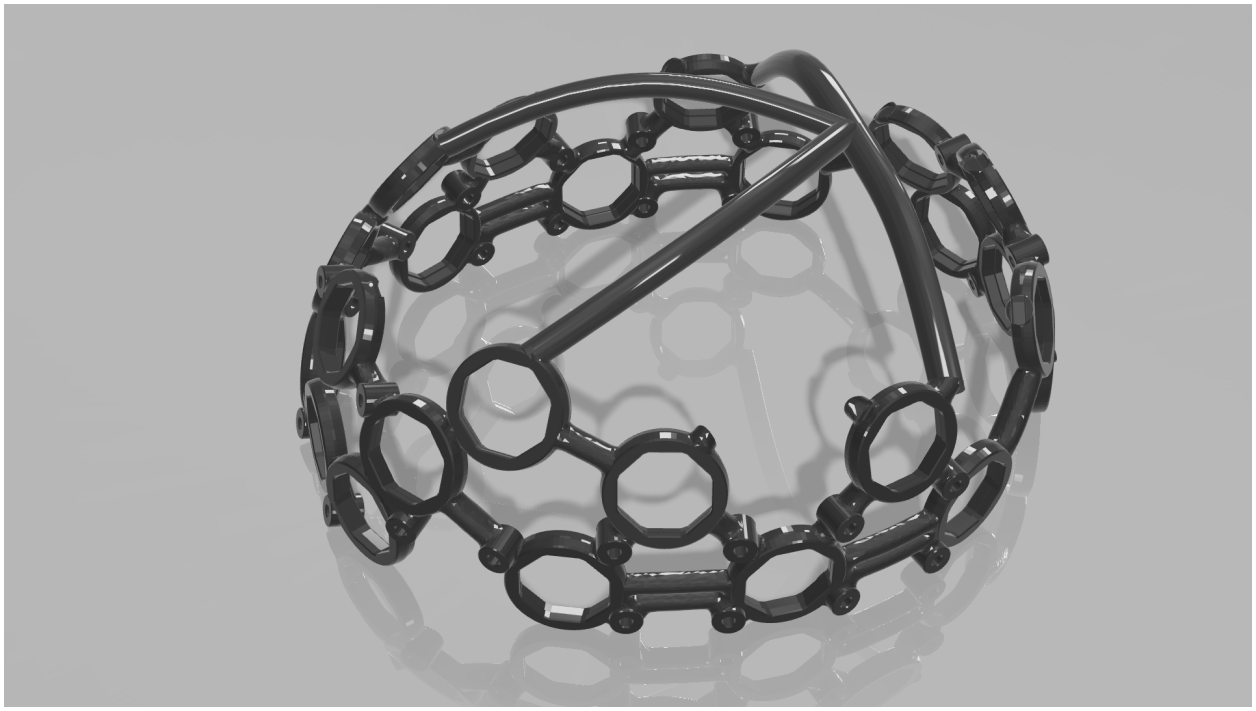
Tiene una excelente accuracy si se encuentra el máximo absoluto justo sobre la frecuencia de estimulación. De lo contrario ya tenemos en mente implementar un sistema de puntos que tenga en consideración los primeros 2 armónicos y también un intervalo con una incertidumbre alrededor de una frecuencia de estimulación.

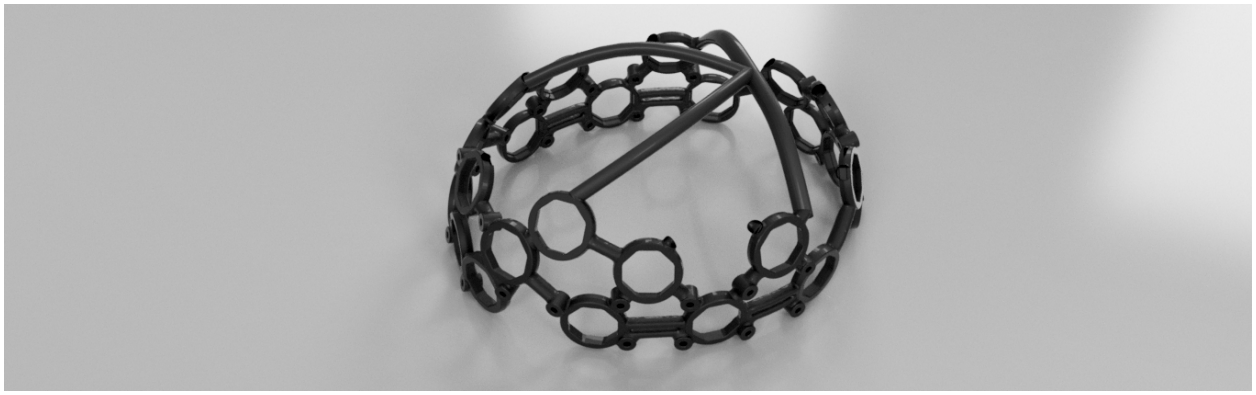
A futuro pensamos en implementarlo a una clase de Python.

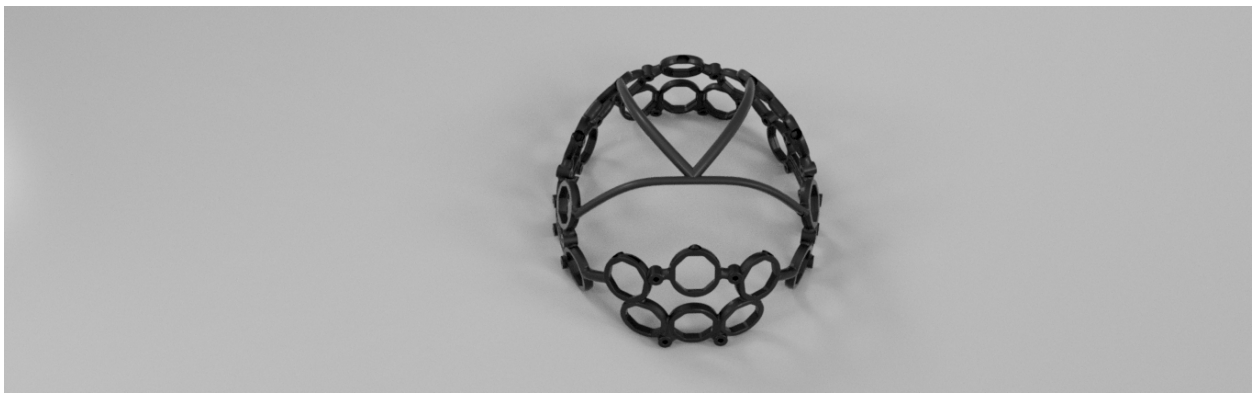
Casco BCI

Realizamos unas modificaciones al diseño del casco para utilizar menos material y que estéticamente tenga el toque de Mentalink. Nuestro objetivo es que al adquirir señales se vea esteticamente el casco diferente que al de la competencia. Los arcos están en esa posición para dar fortaleza al diseño y un toque revolucionario.

Rediseño







Comunicación Bluetooth

En estas últimas dos semanas nos hemos ocupado el M1 a implementar la conexión por medio de BT entre el M1 y el M3, inclusive implementamos un protocolo de comunicación que subimos al repositorio.

Configuración maestro-esclavo

Logramos configurar exitosamente dos módulos BT HC-05, uno como esclavo y otro como maestro.

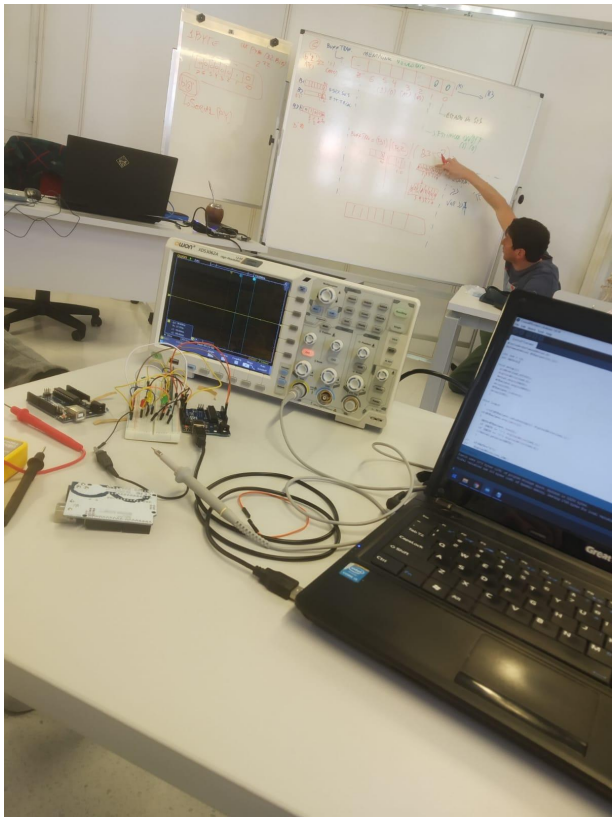
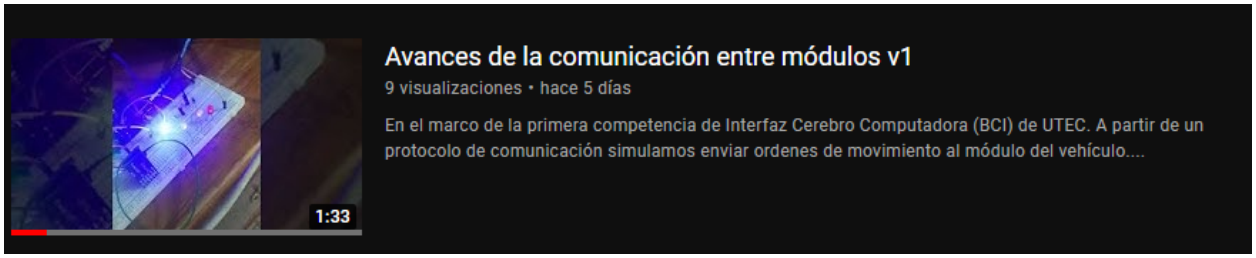


Imagen del viernes 17 de setiembre en UTEC, donde logramos implementar en un día el envío de varios bytes entre Arduinos.

Envío de datos

1- En conjunto con el protocolo de comunicación que redactamos implementamos primero el envío de bytes desde Arduino del M1 y luego hacia el M3. Encendemos y apagamos un LED cuando lograban establecer la comunicación, pero esta orden se la daba en todo momento el M1 Maestro al M3 esclavo https://www.youtube.com/watch?v=T1BrY0m8kiY



2- Luego establecimos la comunicación entre Python hacia el Arduino del M1 y luego hacia el M3. Sin mencionar que cambiamos la frecuencia de oscilación de los LEDs de estimulación simulando una instancia real donde se estimula, clasifica y envía la orden de detenerse.

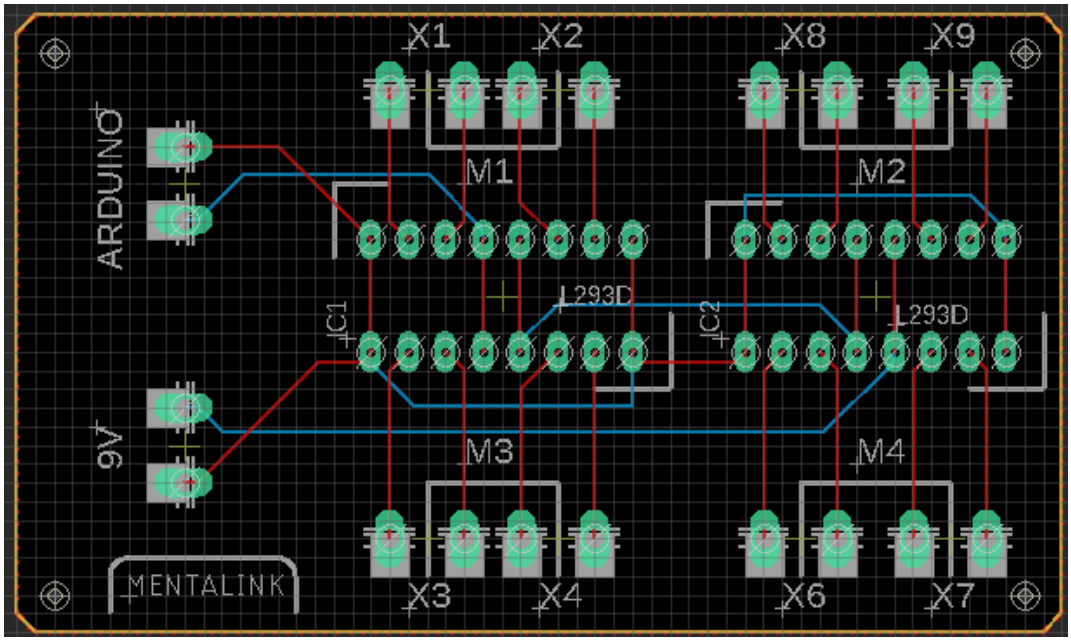
Como próximos retos:

3- Implementamos el enmascaramiento de bytes provenientes de Python en el M1 y lo enviamos al M3 para que haga determinada orden (LED asignado). Cuando el auto no se esté moviendo estas LEDs se apagan simbolizando que terminó el movimiento del robot, y que coincide con el tiempo que las LEDs de estimulación están prendidas.

4- Implementación del apagado de LEDs de estimulación si el auto no se puede mover a determinada dirección.

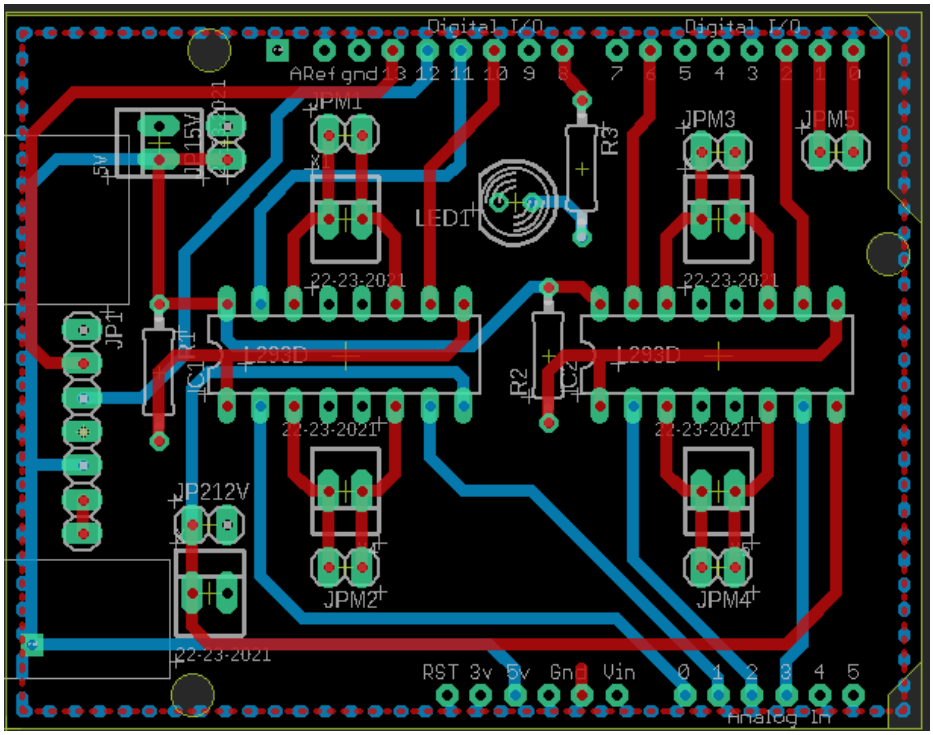
Vehículo

Diseño de shield para motores:
Al inicio se creó esta pcb que iba a encontrarse separado del arduino. Luego de crear la disposición y el diseño ,Lucas Baldezzari nos sugirió aumentar el ancho de las pistas .

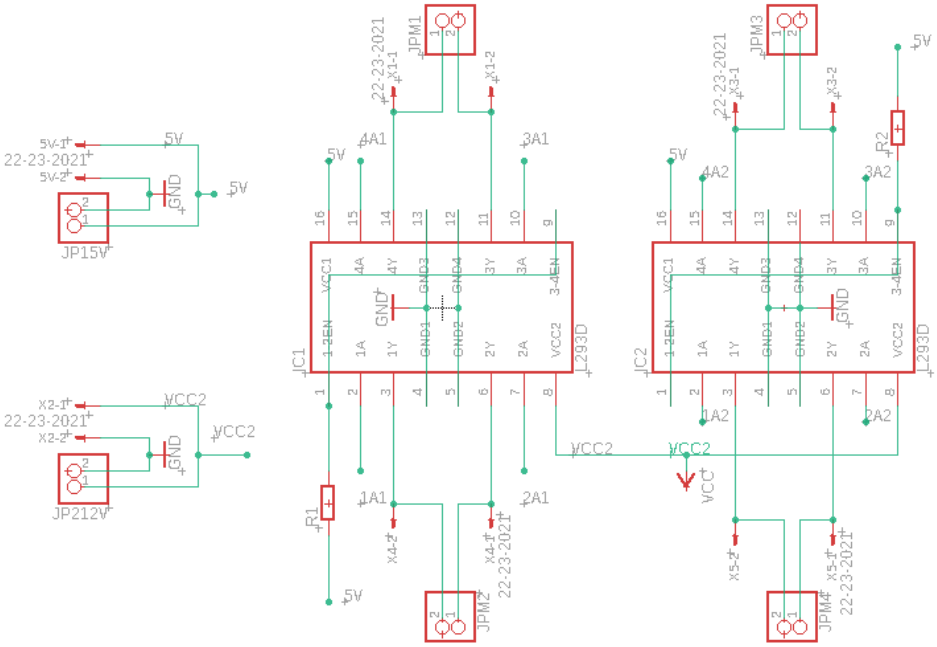


A lo que al tiempo siguiendo sus sugerencias se llega a la conclusión de utilizar una shield para el control de los motores.

Shield:



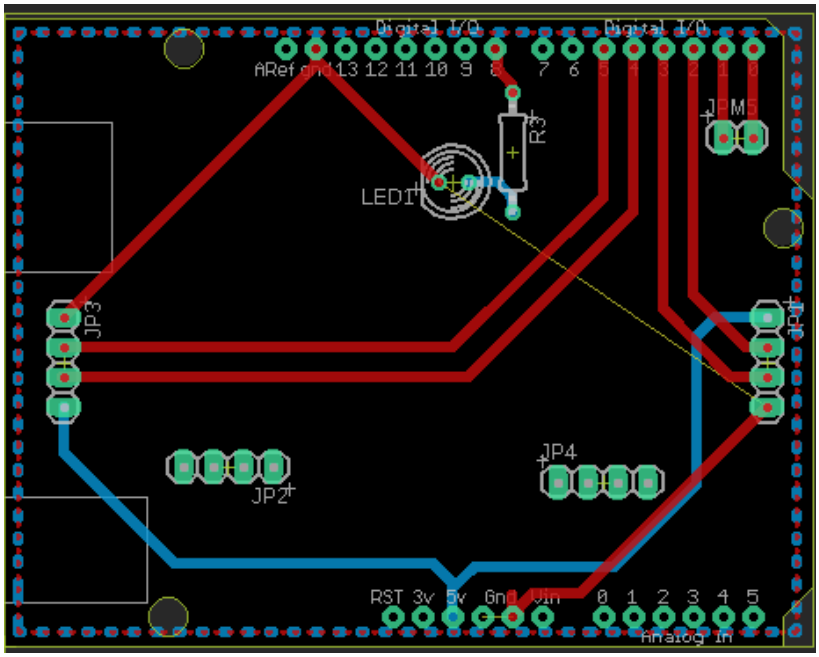
Esquemático:



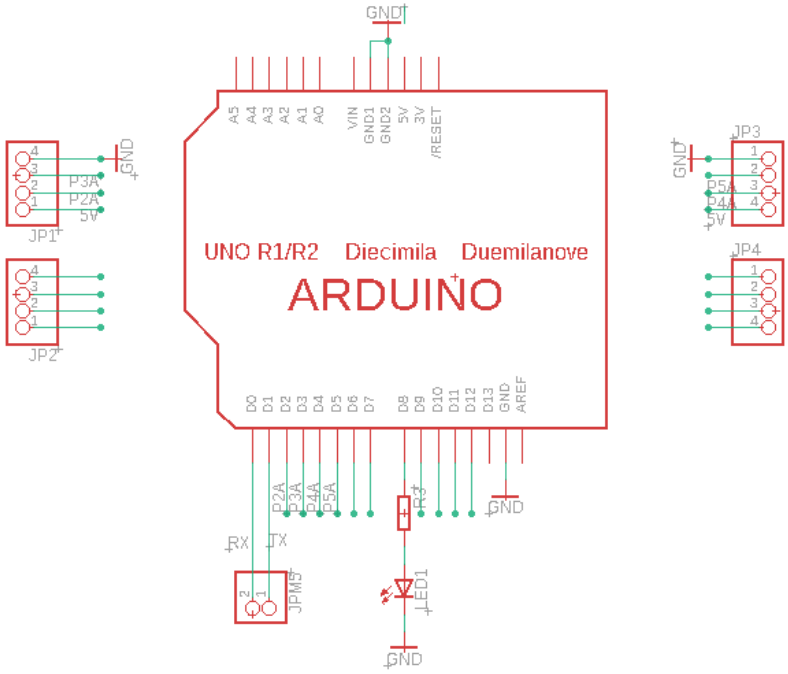
Esquemático de las conexiones de la shield en el arduino :

Diseño de shield para los sensores: *Está shield aun no está terminada ya que estamos evaluando la mejor composición y diseño de la misma. A su vez primero probaremos la funcionalidad de la primera shield, además de esto primero queremos ver el comportamiento del vehículo con todo los componentes y posiblemente agregar más componentes en la segunda pcb.*

Shield sensores:



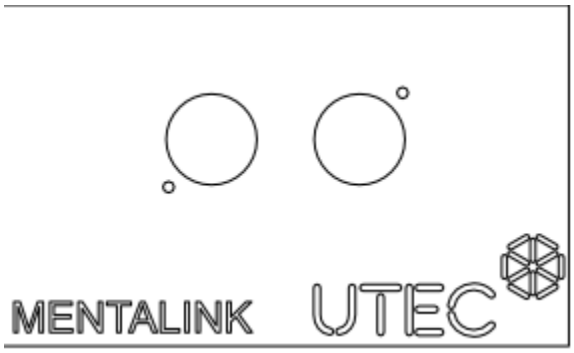
Esquemático:



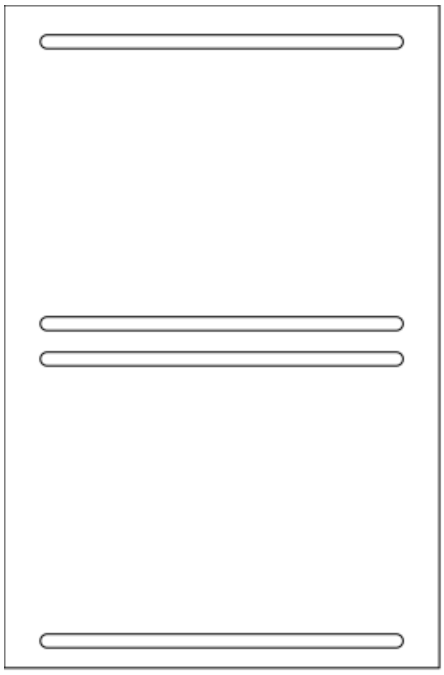
A raíz de que tuvimos que hacer unos cambios en las conexiones de las placas también se vio afectado el código que ya teníamos creado en tinkercad, este mismo lo tuvimos que modificar ya que es más sencillo hacer cambios en el Firmware que en el Hardware

Diseños de archivos para los cortes del acrílico:

Aquí podemos observar la parte delantera que se diseñó con orificios para el sensor de ultrasonido (HC-SR04) el cual estará atornillado en acrílico.



intentamos buscar una forma óptima para tener nuestro cableado y a la vez que quedara elegante a la vista de las personas, para ello se decidió hacer una especie de mesa la cual tendrá todo los componentes sobre ella y su cableado pasará por los orificios que se pueden apreciar en la siguiente foto.



También se diseñó el archivo para unir los motores con las piezas de acrílico el cual se pondrá a prueba y se verá su eficiencia al momento de aguantar el peso de los componentes y las fuerzas que ejercen las ruedas