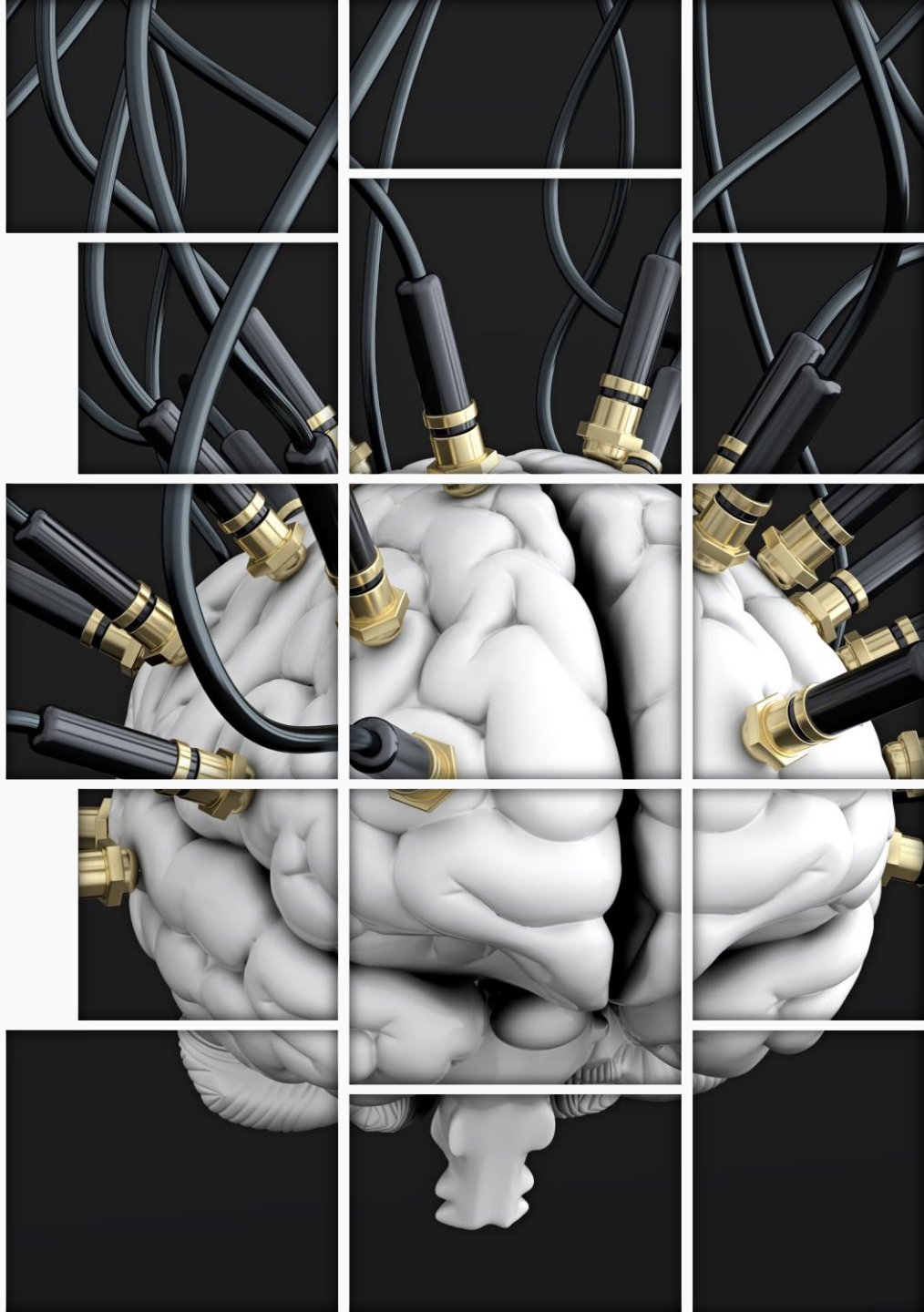
UTEC
Universidad Tecnológica

Taller N°5

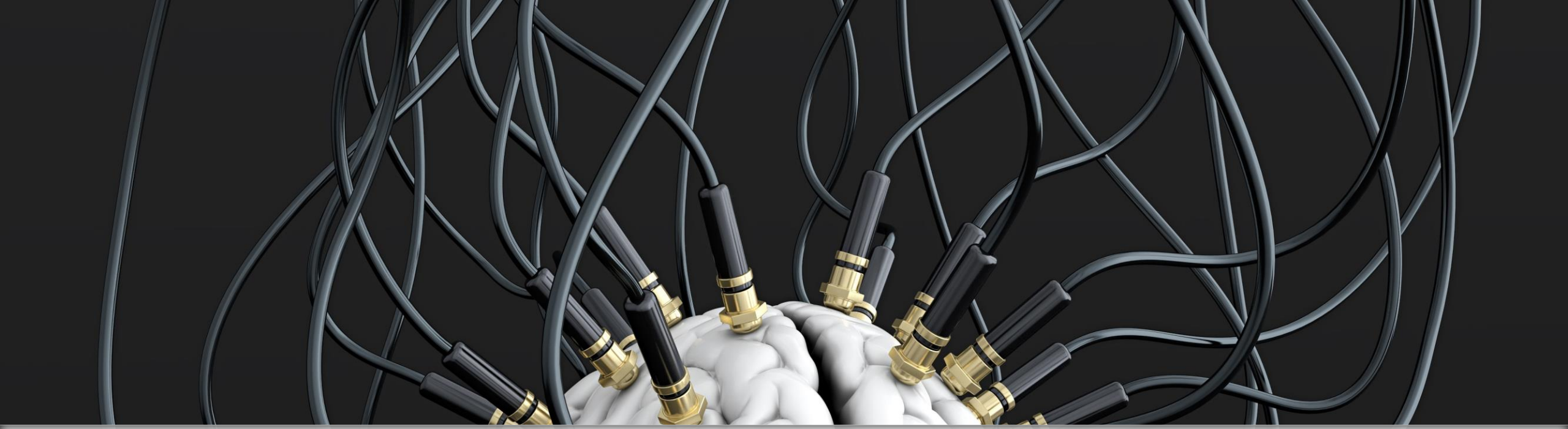
Comunicación serie entre PC y Arduino con Python

Mgr. Bioing. Baldezzari Lucas
Profesor Encargado
Ingeniería Biomédica



Fray Bentos, Uruguay

25 de Agosto de 2021

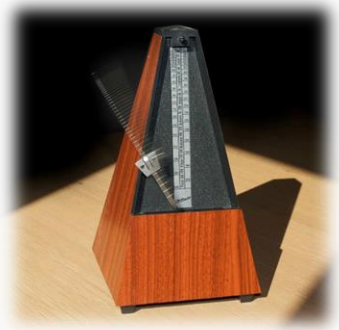


Objetivos del taller

- Entender al Módulo 1 cómo el administrador de procesos.
- Comunicar PC y Arduino mediante Python a través del puerto serie de la PC.
- Utilizar la comunicación serie para sincronizar estímulos y recepción/envío de información.
- Comunicación entre Python y sitio en HTML para sincronizar estímulos.
- Manejo de versiones en archivos de firmware y hardware.

¿Por qué el M1 será el administrador de procesos?

M1 – El director de Orquesta



La importancia de gestionar tareas

En este proyecto, el M1 tiene las siguientes tareas fundamentales,

- Estimular a la persona para evocar SSVEPs.
- Adquirir y procesar señales de EEG de manera *offline* para entrenar clasificadores.
- Adquirir y procesar la señal de EEG en tiempo real para obtener un comando que será enviado al M3.

Pero también,

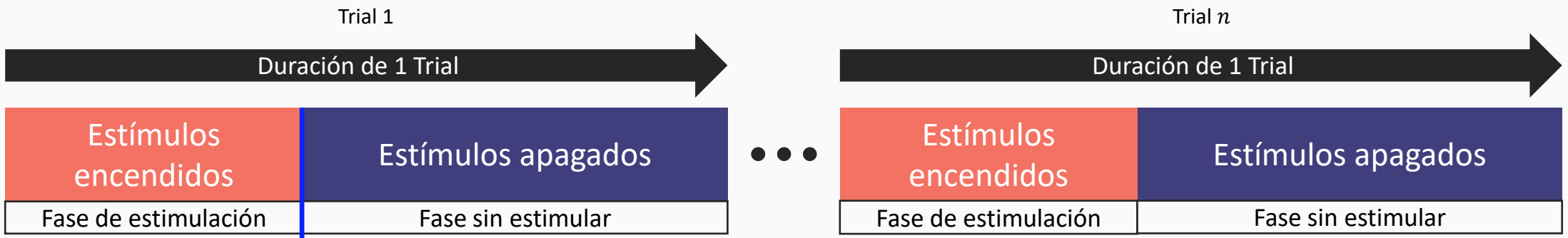
Es quien estará a cargo de sincronizar todas las tareas de la BCI.

Nuestro metrónomo serán los Trials.

M1 – El director de Orquesta

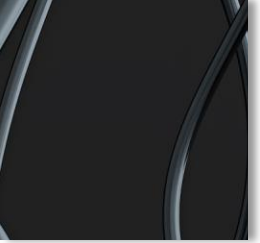
La importancia de gestionar tareas

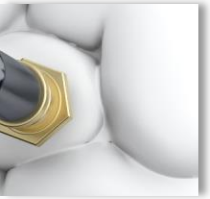
Nuestro metrónomo serán los Trials.

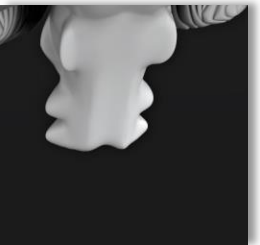


Al finalizar cada fase de estimulación realizaremos varias cosas, entre estas,

1. Adquirir un “pedazo” de la señal de EEG igual al tiempo de la fase de estimulación.
2. Procesar y clasificar la señal de EEG para obtener un comando.
3. Enviar datos al Arduino M1 con el comando obtenido -y de ahí al Arduino M3-. Recibir datos del estado del robot desde Arduino M1 –que fueron enviados primero desde Arduino M3-.
4. Graficar señal de EEG y mostrar algunos datos relevantes en pantalla.





M1 – El director de Orquesta

Similitudes y Diferencias entre Mentalink y Neurorace

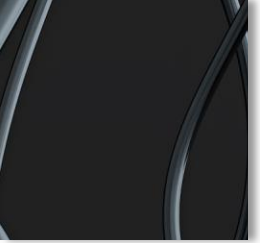
Similitudes más importante:

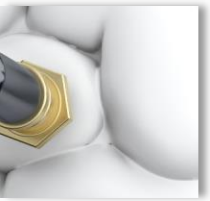
- Adquisición, registro y procesamiento de EEG en **Python**.
- Administrador de procesos.

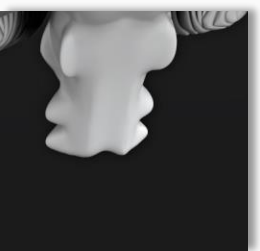
Diferencias

Las diferencias están principalmente en cómo se muestran los estímulos y cómo se envían los comandos al M3.

Veamos…





M1 – El director de Orquesta

Diferencias entre Mentalink y Neurorace

MENTALINK

PC - ADMINISTRADOR DE EVENTOS

- CONTROL DE TRIALS
- CONTROL DE ESTÍMULOS
- PROCESAMIENTO DE EEG.
- OBTENER UN COMANDO

SE ENVÍA

- ESTADO DE LA SESIÓN
- ESTÍMULOS ON/OFF
- COMANDO PARA EL VEHÍCULO.

SE RECIBE

- ESTADO INTERNO DEL ROBOT

¡SERIE!

- PRESENTACIÓN DE ESTÍMULOS
- ENVÍO DE COMANDO AL M3.
- RECEPCIÓN DE ESTADO M3.

ARDUINO M1

¡BLUETOOTH!

ARDUINO M3

- RECEPCIÓN DE COMANDO DEL M1.
- CONTROL MOVIMIENTOS ROBOT.
- MONITOREO DE OBSTÁCULOS
- ENVÍO ESTADO INTERNO AL M1.

M1 – El director de Orquesta

Diferencias entre Mentalink y Neurorace

NEURORACE.

PC - ADMINISTRADOR DE EVENTOS.

- CONTROL DE TRIALS
- CONTROL DE ESTÍMULOS
- PROCESAMIENTO DE EEG.
- OBTENER UN COMANDO
- PRESENTACIÓN DE ESTÍMULOS → (HTML)
- ENVÍO DE COMANDO AL M3. → DESDE PYTHON USANDO WIFI
- RECEPCIÓN DE ESTADO M3. → DESDE PYTHON

- SIMILITUDES
- DIFERENCIAS

ARDUINO M3

- RECEPCIÓN DE COMANDO DEL M1.
- CONTROL MOVIMIENTOS ROBOT.
- MONITOREO DE OBSTÁCULOS
- ENVÍO ESTADO INICIO AL M1.

¡WIFI!

Comunicación serie

Comunicación serie

Entre Arduino y PC

La comunicación serie entre PC y Arduino la haremos con Pyserial.

“This module encapsulates the access for the serial port.” (Documentación)

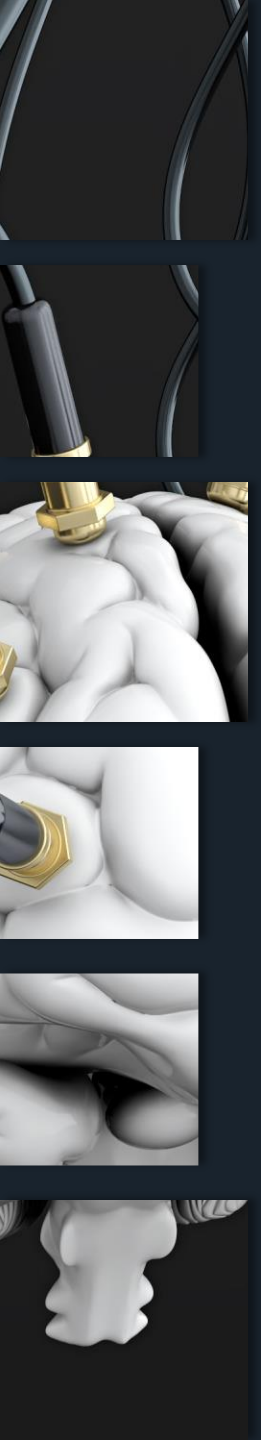
Para poder comunicarnos entre la PC y Arduino vamos a utilizar un script en Python y un firmware en el Arduino M1.

ArduinoCommunication

Clase implementada en Python que nos permitirá comunicarnos con el Arduino M1

Firmware Arduino M1

Nos permitirá enviar comandos al Arduino M3 y recibir comandos desde el. (En el caso de Mentalink les permitirá también el control de los estímulos)



Comunicación serie

Desde Python

Veamos la implementación en Python desde una Jupyter Notebook.

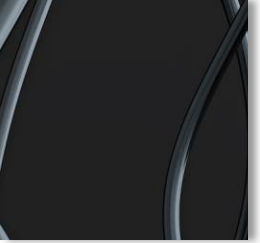
Para ejecutar el ambiente de la notebook de manera local abrimos desde Jupyter desde la app, o desde consola haciendo,

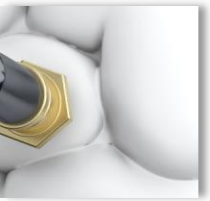
i) conda install -c anaconda ipykernel

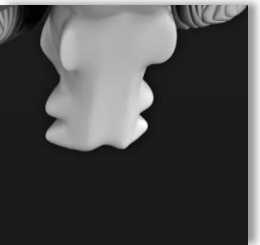
ii) jupyter notebook --notebook-dir "\rutaADirectorio"

Luego abrir la notebook *"ArduinoCommunication.ipynb"*.

También pueden ver la notebook desde el repositorio.





Comunicación serie

Desde Arduino M1

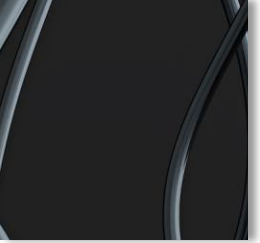
La estructura de archivos para este proyecto es,

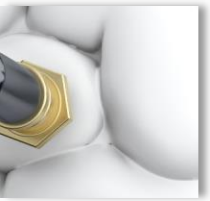
- *inicializaciones.h*: Contiene algunas funciones que se utilizan para inicializar timers.
- *definiciones.h*: Contiene variables definidas mediante directiva #define. Sirven como labels para las variables de estado.
- *main.ino*: Archivo con el programa principal.

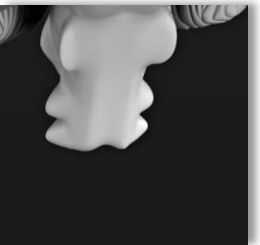
Abrir la notebook *"ArduinoM1Firmware.ipynb"*

```
104  void stimuliControl()
105  {
106    acumuladorStimuliON++;
107    switch(stimuli)
108    {
109      case ON:
110        //control estímulo izquierdo
111        if (++acumEstimIzq >= estimIzqMaxValue)
112        {
113          estimIzqON = !estimIzqON;
114          digitalWrite(estimIzq,estimIzqON);
115          acumEstimIzq = 0;
116        }
117
118        //control estímulo derecho
119        if (++acumEstimDer >= estimDerMaxValue)
120        {
121          estimDerON = !estimDerON;
122          digitalWrite(estimDer,estimDerON);
123          acumEstimDer = 0;
124        }
125        break;
126
127      case OFF:
128        {
129          //Apagamos estímulos y reiniciamos contadores
130          digitalWrite(estimIzq,0);
131          acumEstimIzq = 0;
```

Python y HTML





Comunicación con HTML

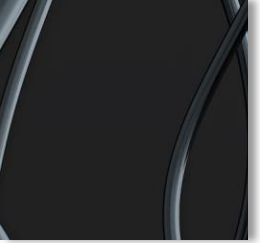
Python haciendo de servidor

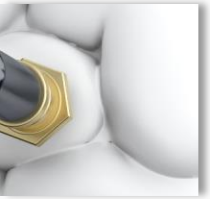
Para poder comunicarnos con un cliente HTML desde Python debemos emular un Servidor. Esto lo hacemos con el paquete *Flask* (docs.).

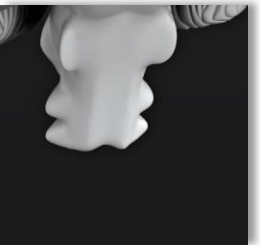
Importante: El control de los estímulos sobre la pantalla tiene **la misma lógica** que vimos anteriormente. La diferencia es que ahora **no le enviamos datos de control de estímulo al Arduino M1.**

A través de *sockets* actualizaremos los estímulos en el *cliente*.

Control de versiones





Controlando las versiones de nuestros códigos

Codificación

Tres primeras letras indican qué representa el archivo.

- FMR = Firmware (para códigos en Arduino).
- SCT = Script (para códigos en Python).
- M3D = Modelo 3D para imprimir.
- PLA = Placa electrónica (board y esquemático).

Los dos números siguientes indican la versión.

- 01, 02, …

Le sigue la palabra *Rev* acompañada de una letra en mayúsculas.

- Rev A, Rev B, etc…

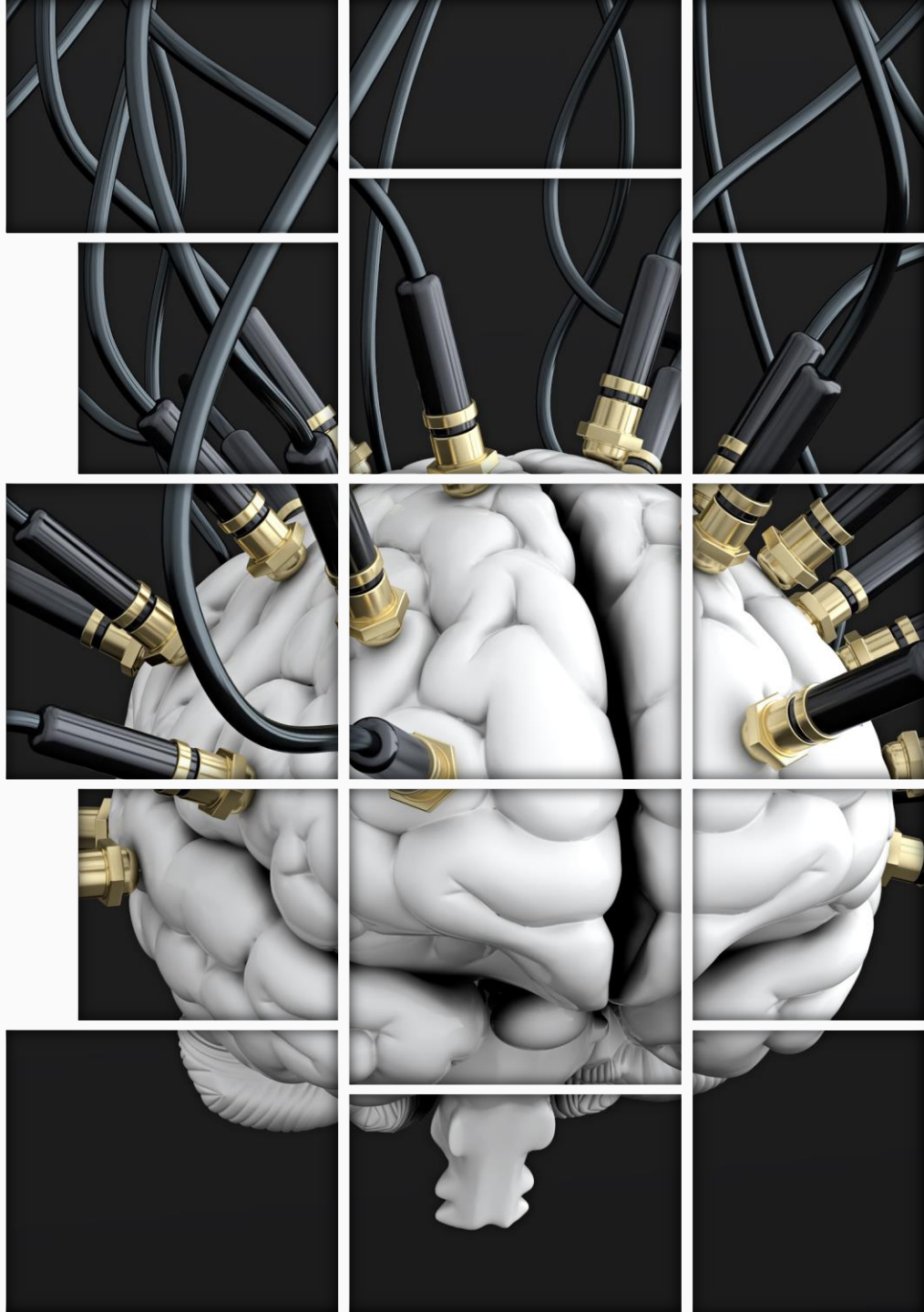
A modo de ejemplo,

FMR-01-Rev A

Representa la versión 1, revisión A de un Firmware de Arduino.

UTEC
Universidad Tecnológica

¿Preguntas?



Fray Bentos, Uruguay

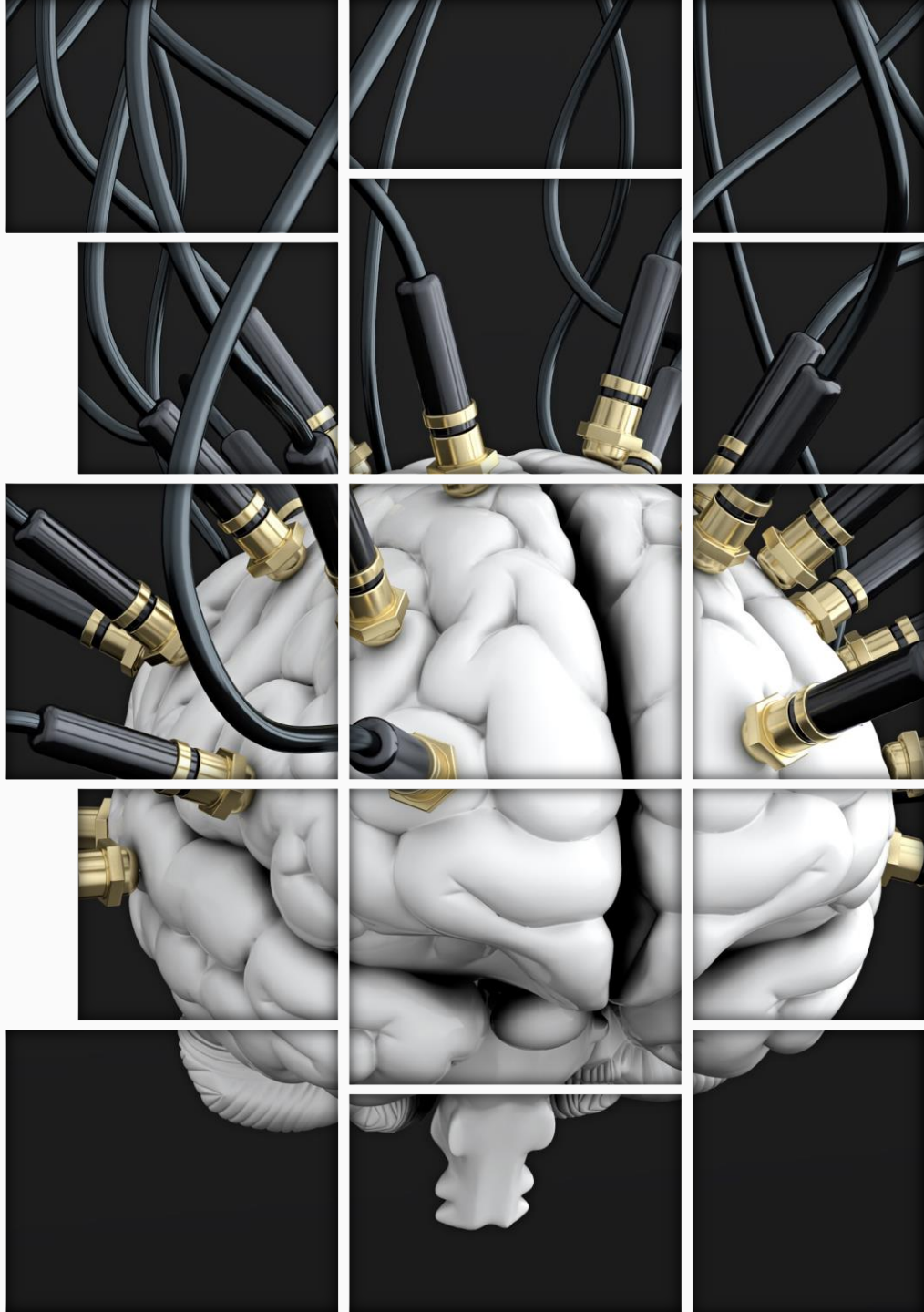
25 de Agosto de 2021

UTEC
Universidad Tecnológica

Taller N°5

Comunicación serie entre PC y Arduino con Python

Mgr. Bioing. Baldezzari Lucas
Profesor Encargado
Ingeniería Biomédica



Fray Bentos, Uruguay

25 de Agosto de 2021