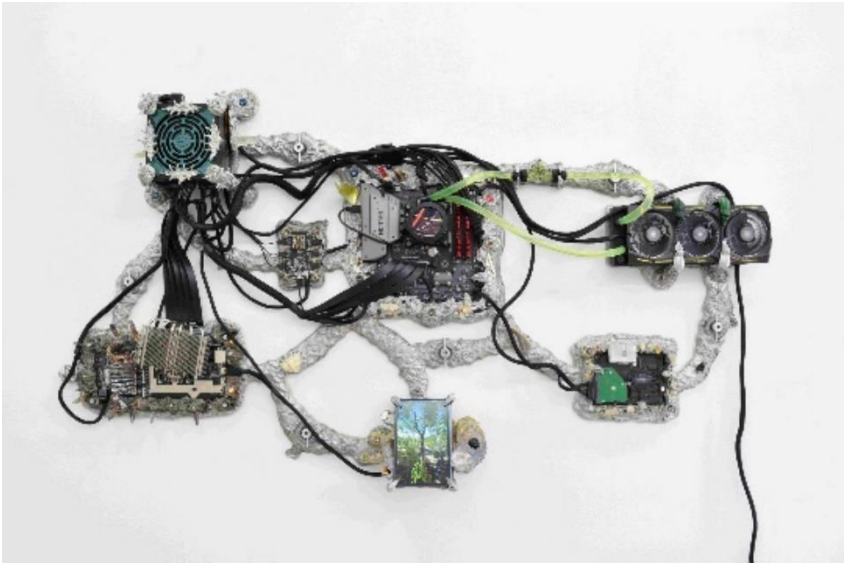
ANEXO I: MEMORIA

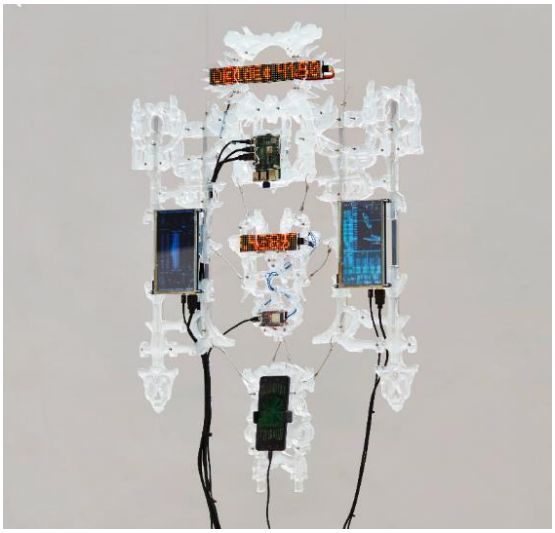
Jenniecam, 1995. Nacimiento del Livecasting.

Case Mode 3, 2023. Janne Schimmel

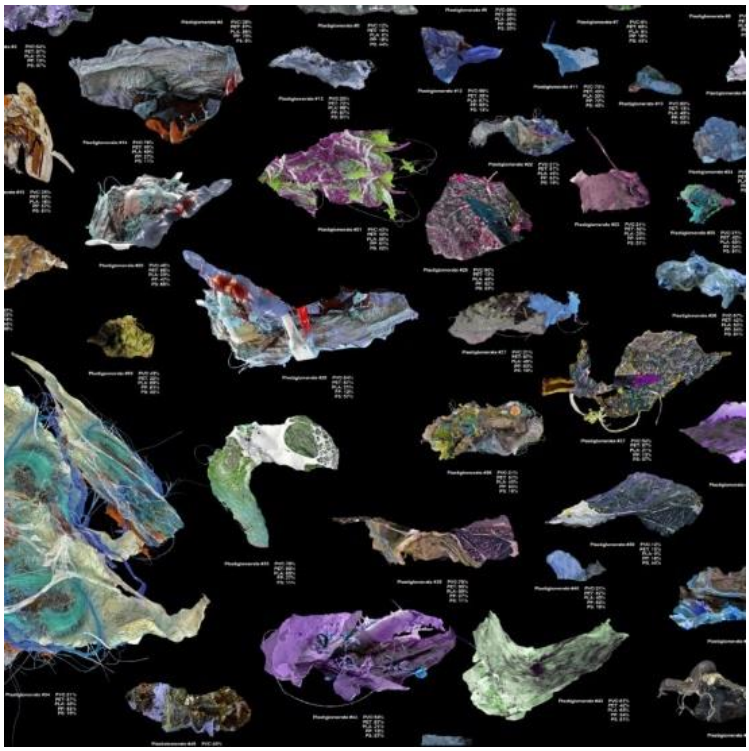
Seeders and Leecher, 2019. Janne Schimmel.



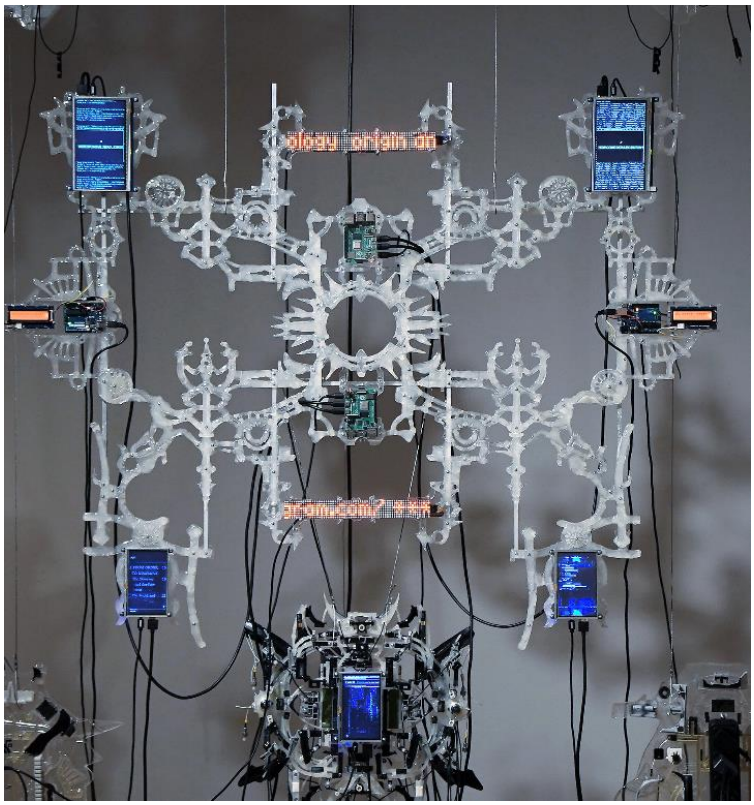
Dictum Terminal, 2023. Teresa Fernández-Pello.



Hiperlung, 2021. Teresa Fernández-Pelló.

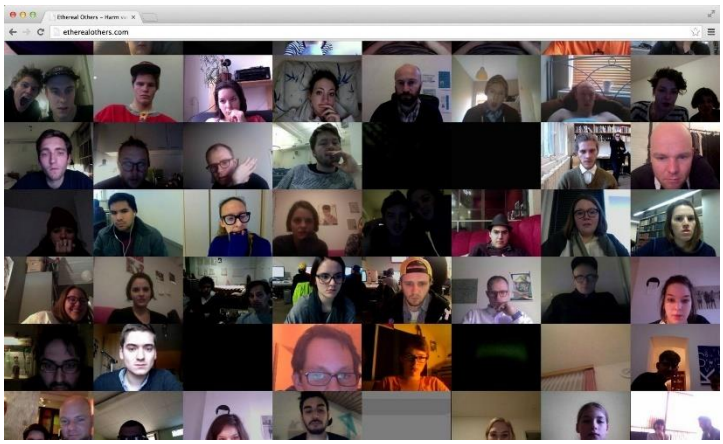


Plastics of the Mediterranean, 2020. Teresa Fernández-Pello y Matteo Guamaccia. Vista Detalle.

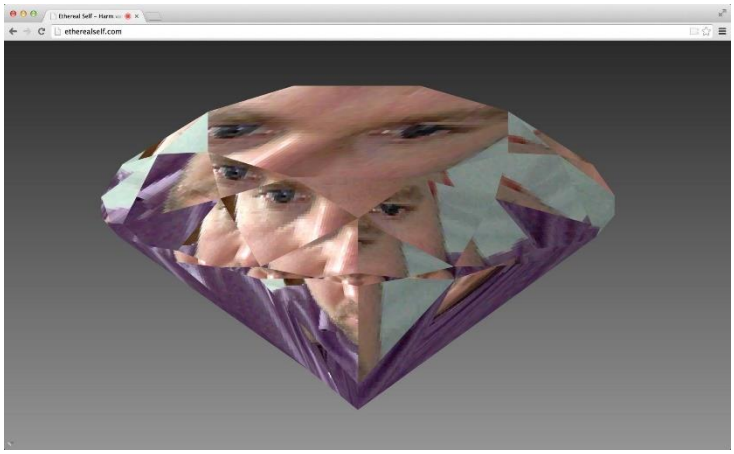


The Heart of the Hurt, 2023. Teresa Fernández-Pello. Vista detalle.

Etheral Others, 2008. Harm van den Dorpel.

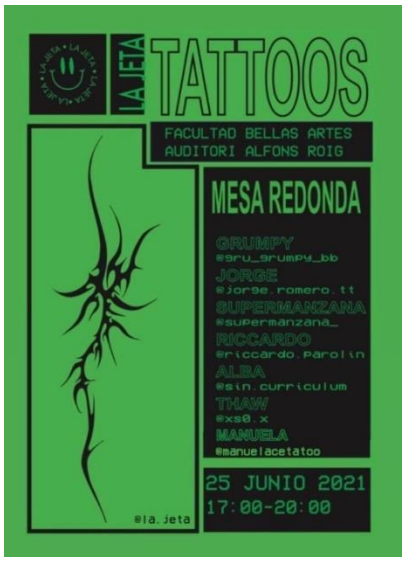


Etheral Self, 2008. Harm van den Dorpel.



Tour Spatiodynamique et Cybernétique, Nicolas Schöffer, 1955

La Jeta: Tatoos, 2020. Cartelería por Alicia Ezpeleta.



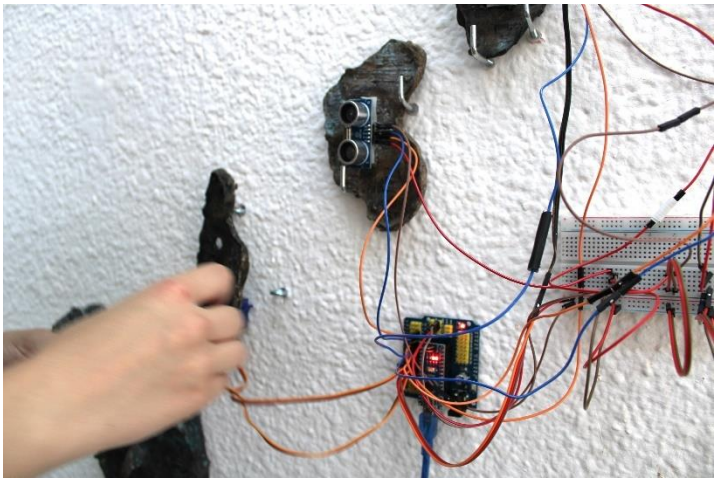
Sara Alastuey, 2023. *Kiati_03.* Fotografía por Margarida Bolsa.

Sara Alastuey, 2023. *Kiati_03.* Vista detalle. Fotografía por Margarida Bolsa.

Sara Alastuey, Alicia Ezpeleta y Javier Terrazas, 2023. *Prótesis*

Sara Alastuey, Alicia Ezpeleta y Javier Terrazas, 2023. *Prótesis*

Sara Alastuey, 2024. *Living as Data.* Vista Detalle. Fotografía por Nerea Moratilla Hijosa.



Sara Alastuey, 2024. *Living as Data.* Vista Detalle. Fotografía por Nerea Moratilla Hijosa.

2024. Muestra Digital Entites. Fotografía por Nerea Moratlla Hijosa.

VIDEOS

Sara Alastuey, 2020. *Present Palace*. Video https://youtu.be/3ac6eCGaObc

Sara Alastuey y Alicia Ezpelta, 2024*. Impuls x La Jeta*. Vídeo https://youtu.be/bmqEuMAWFOM

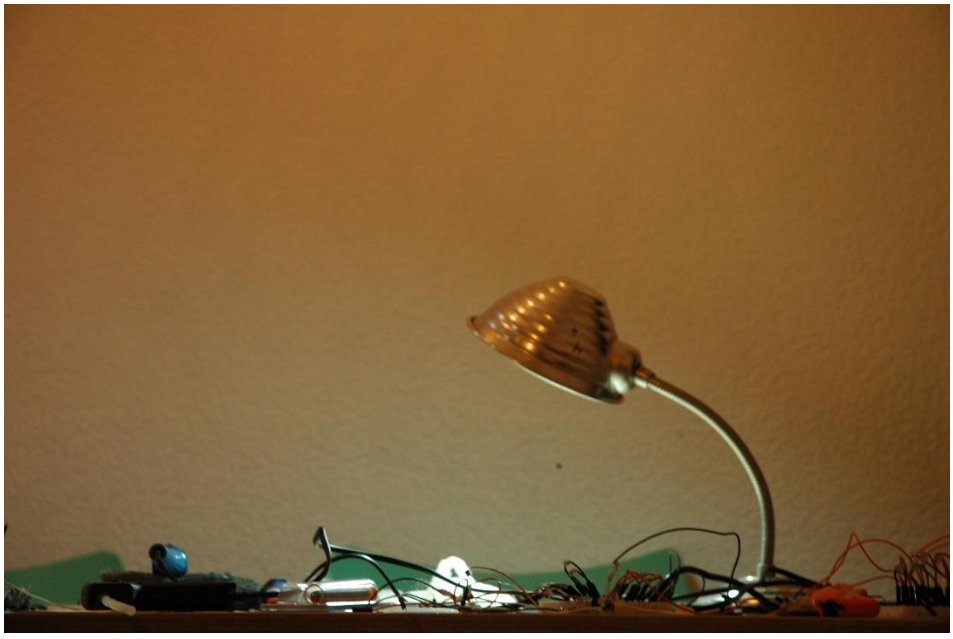
Sara Alastuey, Alicia Ezpeleta y Paula Hernanz, 2024. Prototipo. https://youtu.be/q8Wimr3CgPY

Sara Alastuey y Alicia Ezpeleta, 2024. *Data Driven Dialogs*. https://youtu.be/O-itdkXqB00

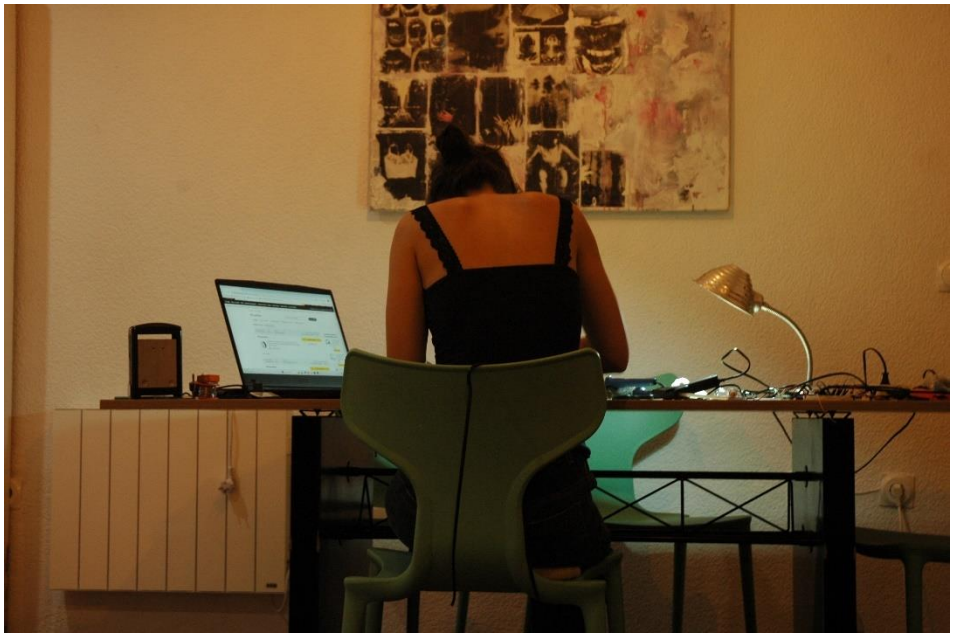
Sara Alastuey, 2024. *Living As Data*. https://youtu.be/7t7THFmVYKw

Sara Alastuey y Alicia Ezpeleta, *Prótesis* 2024 https://youtu.be/tU4mT1Z608I

ABEXO II: PROCESO



2024 Proceso de ensamblaje y soldadura del cableado. *Data Driven Dialogs.* Fotografía por Julia Garcia Artero.

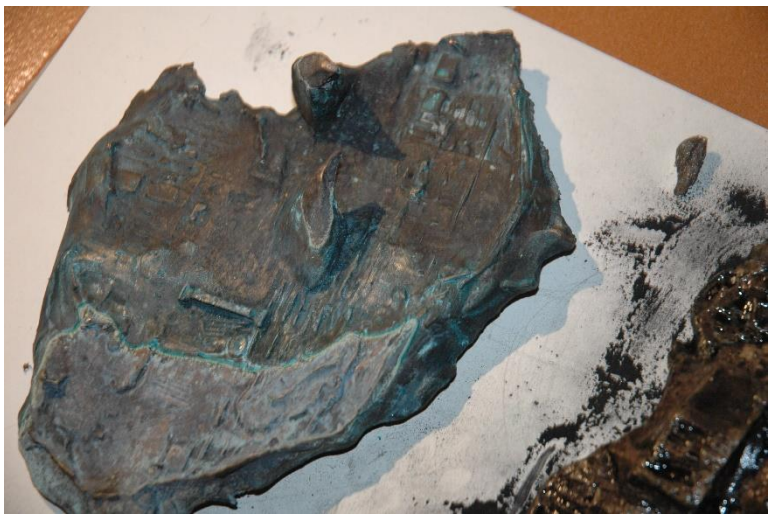


2024 Proceso de ensamblaje y soldadura del cableado. *Data Driven Dialogs.* Fotografía por Julia Garcia Artero.

2024. Muestra *Digital Entites*. Fotografía por Nerea Moratilla Hijosa.

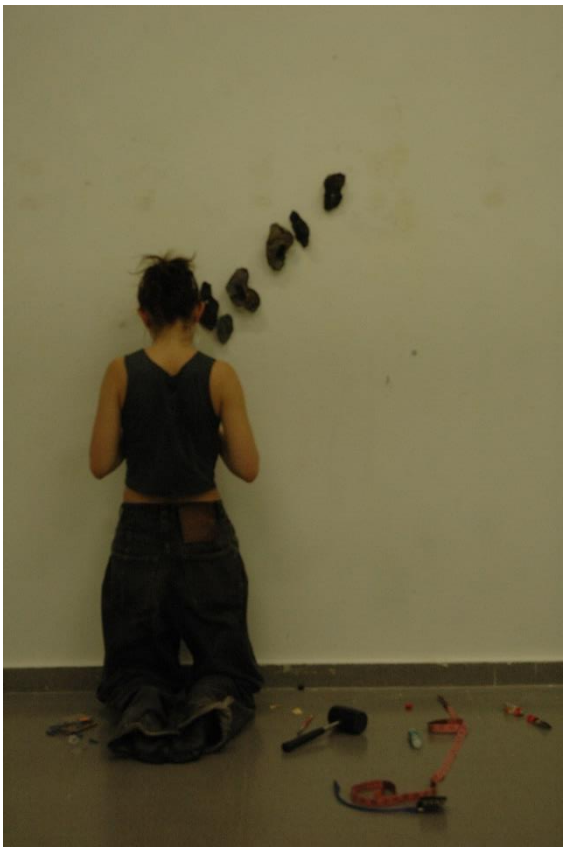
2024 Árbol de colada en bronce. *Living as Data.*

2024 Proceso de patinado en frío. *Living as Data.* Fotografía por Julia García Artero.



2024 Proceso de patinado en frío. *Living as Data.* Fotografía por Julia García Artero.

Proceso de patinado en frío. *Living as Data.* Fotografía por Julia García Artero.



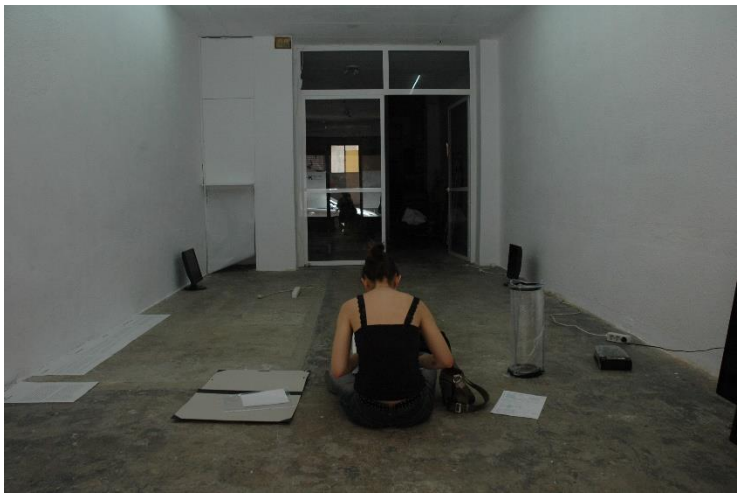
2024 Prueba de instalación en la proyect room de Facultad de San Carlos A-02-9. *Living as Data*
Fotografía por Julia García Artero.

2024 Prueba de instalación en la proyect room de Facultad de San Carlos A-02-9. *Living as Data*
Fotografía por Julia García Artero.

2024 Prueba de instalación en la proyect room de Facultad de San Carlos A-02-9. *Living as Data*
Fotografía por Julia García Artero.

2024 Instalación muestra *Digital Entities. Living as Data* Fotografía por Julia García Artero.

2024 Instalación muestra *Digital Entities* Fotografía por Julia García Artero.



2024 Muestra Digital *Entities. Fotograrafía* por Nerea Moratilla Hijosa.

PROGRAMACIÓN

DATA DRIVEN DIALOGS

```
#define ATMEGA32U4 1
#define DEBUG 1
#include "MIDIUSB.h"
#include <ResponsiveAnalogRead.h>

// ASIGNACIÓN BOTONES
const int numero_botones = 3;
const int pines_botones[numero_botones] = {2, 3, 4, 5};

int boton_valoractual[numero_botones] = {};
int boton_valoranterior[numero_botones] = {};

// DEBOUNCE
unsigned long lastDebounceTime[numero_botones] = {0};
unsigned long debounceDelay = 50;

// ASIGNACIÓN POTENCIÓMETROS
const int N_POTS = 1;

int potPin[N_POTS] = {A1};
int potState[N_POTS] = {0};
int midiState[N_POTS] = {0};
int midiStateAnterior[N_POTS] = {0};  // Agregada para almacenar los valores
anteriores

// MIDI
byte midiCh = 1;  // Puedes ajustar el canal MIDI según tus necesidades
byte note = 36;

void setup() {
  Serial.begin(31250);  // Cambiar baudios a 31250 si MIDI class compliant o de-
jarlos así con Hairless MIDI
  Serial.println("Debug mode");
  Serial.println();

  // BOTONES
  for (int i = 0; i < numero_botones; i++) {
    pinMode(pines_botones[i], INPUT_PULLUP);
  }
}

void loop() {
```

```
 BOTONES();
 POTENCIOMETROS();
}

// BOTONES
void BOTONES() {
 for (int i = 0; i < numero_botones; i++) {
  boton_valoractual[i] = digitalRead(pines_botones[i]);

  if ((millis() - lastDebounceTime[i]) > debounceDelay) {
   if (boton_valoranterior[i] != boton_valoractual[i]) {
    lastDebounceTime[i] = millis();
    if (boton_valoractual[i] == LOW) {
     noteOn(midiCh, note + i, 127);
     MidiUSB.flush();
     Serial.print(i);
     Serial.println(": button on");
    } else {
     noteOn(midiCh, note + i, 0);
     MidiUSB.flush();
     Serial.print(i);
     Serial.println(": button off");
    }
    boton_valoranterior[i] = boton_valoractual[i];
   }
  }
 }
}

// POTENCIOMETROS
void POTENCIOMETROS() {
 const int umbralPot = 5;  // Ajusta este valor según sea necesario
 for (int i = 0; i < N_POTS; i++) {
  potState[i] = analogRead(potPin[i]);
  midiState[i] = map(potState[i], 0, 1023, 0, 127);

  // Filtrar valores pequeños
  if (abs(midiState[i] - midiStateAnterior[i]) > umbralPot) {
   // Enviar mensaje de cambio de control MIDI solo si hay un cambio signifi-
cativo
   controlChange(midiCh, i + 1, midiState[i]);  // El segundo parámetro es el
número de control MIDI, comienza desde 1

   Serial.print("PotState: ");
   Serial.print(potState[i]);
   Serial.print(" midiState: ");
   Serial.println(midiState[i]);
```

```
    // Actualizar el valor anterior solo si se envía un nuevo mensaje MIDI
    midiStateAnterior[i] = midiState[i];
   }
  }
  Serial.println();
}

// MIDIUSB Library
void noteOn(byte channel, byte pitch, byte velocity) {
  midiEventPacket_t noteOn = {0x09, 0x90 | channel, pitch, velocity};
  MidiUSB.sendMIDI(noteOn);
}

void noteOff(byte channel, byte pitch, byte velocity) {
  midiEventPacket_t noteOff = {0x08, 0x80 | channel, pitch, velocity};
  MidiUSB.sendMIDI(noteOff);
}

void controlChange(byte channel, byte control, byte value) {
  midiEventPacket_t event = {0x0B, 0xB0 | channel, control, value};
  MidiUSB.sendMIDI(event);
}
```

LIVNG AS DATA

```
#include <Servo.h>
#include <NewPing.h>
#include <TaskScheduler.h>

Servo servo1;
Servo servo2;
Servo servo3;

const int trigPin1 = 2;
const int echoPin1 = 3;
const int trigPin2 = 4;
const int echoPin2 = 5;
const int trigPin3 = 6;
const int echoPin3 = 7;

NewPing sonar1(trigPin1, echoPin1);
NewPing sonar2(trigPin2, echoPin2);
NewPing sonar3(trigPin3, echoPin3);

Scheduler scheduler;
```

```
// declarar los movimientos
void moverServo1();
void moverServo2();
void moverServo3();

Task taskMoverServo1(1000, TASK_FOREVER, &moverServo1);
Task taskMoverServo2(1000, TASK_FOREVER, &moverServo2);
Task taskMoverServo3(1000, TASK_FOREVER, &moverServo3);

void setup() {
 // servos
 servo1.attach(8);  // Pin 8
 servo2.attach(9);  // Pin 9
 servo3.attach(10); // Pin 10

 // scheduler / tareas
 scheduler.addTask(taskMoverServo1);
 scheduler.addTask(taskMoverServo2);
 scheduler.addTask(taskMoverServo3);

 // Iniciar el planificador
 scheduler.startNow();

 Serial.begin(9600);
}

void loop() {
 scheduler.execute();

 // distancias
 int distancia1 = sonar1.ping_cm();
 int distancia2 = sonar2.ping_cm();
 int distancia3 = sonar3.ping_cm();

 Serial.print("Distancia 1: ");
 Serial.print(distancia1);
 Serial.print(" - Distancia 2: ");
 Serial.print(distancia2);
 Serial.print(" - Distancia 3: ");
 Serial.println(distancia3);
}

void moverServo1() {
 int distancia = sonar1.ping_cm();
 if (distancia <= 20) {
  servo1.write(0); // Mover el servo 1 a la posiciÃ³n 0 grados
 } else if (distancia <= 40) {
  servo1.write(90); // Mover el servo 1 a la posiciÃ³n 90 grados
```

```
  } else {
    servo1.write(180); // Mover el servo 1 a la posiciÃ³n 180 grados
  }
}

void moverServo2() {
  int distancia = sonar2.ping_cm();
  if (distancia <= 20) {
    servo2.write(0); // Mover el servo 2 a la posiciÃ³n 0 grados
  } else if (distancia <= 40) {
    servo2.write(90); // Mover el servo 2 a la posiciÃ³n 90 grados
  } else {
    servo2.write(180); // Mover el servo 2 a la posiciÃ³n 180 grados
  }
}

void moverServo3() {
  int distancia = sonar3.ping_cm();
  if (distancia <= 20) {
    servo3.write(0); // Mover el servo 3 a la posiciÃ³n 0 grados
  } else if (distancia <= 40) {
    servo3.write(90); // Mover el servo 3 a la posiciÃ³n 90 grados
  } else {
    servo3.write(180); // Mover el servo 3 a la posiciÃ³n 180 grados
  }
}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

FACULTAT DE BELLES ARTS DE SANT CARLES

ANEXO I.

RELACIÓN DEL TRABAJO CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE DE LA AGENDA 2030

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster: Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030.

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**		X		
ODS 4. **Educación de calidad.**			X	
ODS 5. **Igualdad de género.**	X			
ODS 6. **Agua limpia y saneamiento.**			X	
ODS 7. **Energía asequible y no contaminante.**			X	
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades.**	X			
ODS 11. **Ciudades y comunidades sostenibles.**		X		
ODS 12. **Producción y consumo responsables.**			X	
ODS 13. **Acción por el clima.**			X	
ODS 14. **Vida submarina.**	X			X
ODS 15. **Vida de ecosistemas terrestres.**		X		X
ODS 16. **Paz, justicia e instituciones sólidas.**		X		
ODS 17. **Alianzas para lograr objetivos.**	X			

Descripción de la alineación del TFG/TFM con los ODS con un grado de relación más alto.

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster:

Relación del trabajo con los Objetivos de Desarrollo

Este Trabajo de Fin de Grado funciona como un replanteamiento o cuestionamiento de la brecha de posibilidades y oportunidades entre grupos sociales; manteniendo relación con los objetivos de Igualdad de género (ODS 5), Salud y bienestar (ODS 3) y la Reducción de desigualdades (ODS 10).

De la misma forma, replantea y cuestiona los planes de estudio en las universidades hacia una mayor relación con los planteamientos interdisciplinares que surgen en contextos externos a la Academia, pero presentes en las instituciones, para asegurar así una Educación de calidad (ODS 4) y una situación de Paz, justicia e instituciones sólidas (ODS 16).

Por otra parte, la realización de una producción artística basada en la reutilización de medios materiales, así como la concepción de obras sin estrictas necesidades de materialización aboga por el desarrollo de los objetivos en torno a la Industria, innovación e infraestructuras (ODS 9); el uso de Energía asequible y no contaminante (ODS 7), y la búsqueda por una Producción y consumo responsables (ODS 12) que aseguren el desarrollo de Ciudades y comunidades sostenibles (ODS 11).