UNIVERSITAT POLITÈCNICA DE VALÈNCIA

OFICIAL^PÚBLICO MÁSTER ARTES VISUALES + MULTIMEDIA

MÁSTER ARTES VISUALES Y MULTIMEDIA
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Trabajo Final de Máster

Redes para una Sociedad de Supervivencia. Proyecto de instalación interactiva auto-organizativa

Presentado por: Manuel Genovés Monzó
Dirigido por: Ma. José Martínez de Pisón Ramón

Valencia, septiembre 2021

Índice

Resumen

El objetivo del presente trabajo es el diseño y planteamiento de la instalación artística Emergencia, cuyos mecanismos de funcionamiento buscan emular los comportamientos emergentes que surgen cuando un conjunto de elementos interaccionan entre ellos, concretamente en el terreno de la biología. Los sistemas simples en cuestión son en este caso pequeños artrópodos electrónicos que interaccionan y modulan su comportamiento basándose en la información que reciben de su entorno cercano, expresando sus estados internos a través de pequeñas luces, y compartiendo información entre ellos a través de una red visible de cobre que los conecta.

De esta manera un conjunto de piezas a priori independientes se comportan como un único ente, actuando de forma individual pero mostrando comportamientos gregarios. La interdependencia así visualizada llama a reflexionar sobre los modelos de sociedad a los que aspiramos como animal social que somos, reflexión especialmente necesaria en un momento de la historia en que somos especialmente vulnerables, y en el que se está polarizando la opinión política precisamente en torno a esta cuestión.

Palabras-clave: COMPORTAMIENTO EMERGENTE, ANARQUISMO, INSTALACIÓN ARTÍSTICA, RED, INTERACCIÓN

Resum

L'objectiu del present treball és el disseny i plantejament de la instal·lació artística Emergència, els mecanismes de funcionament de la qual busquen emular els comportaments emergents que sorgeixen quan un conjunt d'elements interaccionen entre ells, concretament en el terreny de la biologia. Els sistemes simples en qüestió són en aquest cas xicotets artròpodes electrònics que interaccionen i modulen el seu comportament basant-se en la informació que reben del seu entorn pròxim, expressant els seus estats interns a través de xicotetes llums, i compartint informació entre ells a través d'una xarxa visible de coure que els connecta.

D'aquesta manera un conjunt de peces a priori independents es comporten com un únic ens, actuant de manera individual però mostrant comportaments gregaris. La interdependència així visualitzada crida a reflexionar sobre els models de societat als quals aspirem com a animal social que som, reflexió especialment necessària en un moment de la història en què som especialment vulnerables, i en el qual s'està polaritzant l'opinió política precisament entorn d'aquesta qüestió.

Paraules-clau: COMPORTAMENT EMERGENT, ANARQUISME, INSTAL·LACIÓ ARTÍSTICA, XARXA, INTERACCIÓ

Abstract

The objective of this work is the design of the artistic installation Emergence, whose functioning mechanisms seek to emulate the emergent behaviors that arise when a set of elements interact with each other, specifically in the field of biology. The simple systems in question are in this case small electronic arthropods that interact and modulate their behavior based on the information they receive from their immediate environment, expressing their internal states through small lights, and sharing information among themselves through a visible copper network that connects them.

In this way a set of a priori independent pieces behave as a single entity, acting individually but showing gregarious behaviors. The interdependence thus visualized calls us to reflect on the models of society to which we aspire as the social animal that we are, a reflection that is especially necessary at a time in history when we are particularly vulnerable, and in which political opinion is polarizing precisely around this issue.

Keywords: EMERGENCE, ANARCHISM, ART INSTALLATION, NET, INTERACTION

1. Introducción

La historia de la ciencia y la filosofía es la historia del estudio de la complejidad, en tanto en cuanto el mundo que habitamos es un sistema complejo. Desde el nacimiento de la filosofía de la ciencia se establecen dos actitudes ante la complejidad claramente diferenciadas: el holismo y el reduccionismo (Estany, 2006). El holismo entiende que los sistemas complejos hay que estudiarlos como un todo, y quedó relegado a lo largo de la historia a un segundo puesto, tomando la mayoría de las ciencias un enfoque reduccionista en sus metodologías. No obstante la comunidad científica no era ciega al hecho de que en las interacciones entre sistemas se producen fenómenos que dichos sistemas por sí mismos no pueden explicar. Tomando el símil con los niveles de explicación que se suelen referenciar en el área de la psicología cognitivo-conductual (Colombo & Knauff, 2020) junto con el desarrollo de las diferentes áreas del conocimiento se empezaría a intuir que los puentes que las unían no podían explicarse sin la introducción de factores relacionales. Así pues la química sería física aplicada, teniendo en cuenta las interacciones entre muchos elementos. La biología, química aplicada cuando aumentas la complejidad de las cadenas de carbono y de las relaciones entre ellas, y así sucesivamente con todos los campos del saber. Existen tres eventos a destacar que fueron determinantes para que esta aproximación ganase en relevancia: el nacimiento de la termodinámica, la física estadística, y la teoría de la evolución de Darwin. En el libro *Hydrodinamica*, Daniel Bernoulli (1738) introdujo el primer modelo kinético de los gases, estableciendo así el germen de toda una rama de la física en la que las interacciones entre partículas no son una mera consecuencia insustancial a la física que las gobierna, sino una pieza indispensable en toda una categoría de fenómenos emergentes. Se había descubierto el puente que une la física de partículas y la física macroscópica. Un siglo más tarde, un naturalista inglés ultimaba un libro que le haría mundialmente famoso: *El origen de las especies*, (1871) de Charles Darwin (Darwin et al., 2013).

Darwin propuso una hipótesis que revolucionó el mundo científico: un modelo que explicaba cómo las especies evolucionan a lo largo del tiempo, modelo en que los ingredientes principales —tiempo, genes, azar, supervivencia— no tenían ningún tipo de implicación sociológica (si bien el propio Darwin profundizó en el tema en “El origen del Hombre” (Darwin, 1871). ¿Cuál es el factor determinante en la supervivencia de la especie? Toda una escuela de pensamiento señalaría al individuo, escuela de pensamiento que cuenta con tal vigencia que aún hoy se escucha la frase “supervivencia del más fuerte/hábil/adaptado/inteligente” asociada a la teoría de la evolución. Lo cierto es que es una frase que Darwin nunca pronunció, y su pensamiento y posteriores estudios apuntan en una dirección completamente opuesta, y que estudió en mayor detalle Piotr Alekséyevich Kropotkin. En su libro *El apoyo mutuo* (Kropotkin et al., 2016) teoriza sobre las redes de apoyo en las sociedades animales como factor clave en el éxito de una especie, contraponiéndose así a la lectura salvajista de Huxley (pero sin caer en el buenismo de Rousseau). De esta manera se hace patente que los fenómenos emergentes son parte esencial de las sociedades animales; los comportamientos comunitarios que surgen de la relación entre los diversos individuos son un factor clave de la supervivencia de éstos, y adquieren toda una nueva dimensión que supera la suma de los comportamientos individuales de cada uno de ellos. La motivación principal del presente trabajo es el ánimo de estudiar y traer al espacio expositivo este tipo de fenómenos emergentes en el área de la biología , estableciendo —mediante la planificación de un proyecto de instalación interactiva basada en elementos electrónicos interdependientes y redes de cooperación visibles— una suerte de pseudo-sociedad en la que el papel del visitante-observador desaparece para dar paso al del visitante-elemento de interacción.

Un hecho diferenciador con respecto a estudios previos será no obstante que el modelo, además de físico, será ciertamente descentralizado, siendo cada uno de los elementos verdaderamente independientes del resto y obedeciendo a interacciones reales con el entorno. Estudios previos han tendido al uso de simulaciones por ordenador y/o modelos físicos centralizados en los que todos los elementos están controlados desde una sola unidad (Holland, 1998, p. 10). El

estar planteado de esta manera, y, sobre todo, como un entorno abierto en el que los visitantes forman parte indisoluble de la instalación, se busca visibilizar el orden espontáneo como forma válida de auto-organización, siguiendo la máxima popular: lo que no se conoce no existe. El hecho de ser un entorno abierto y no controlado abre la puerta no obstante al fracaso de la tesis que este proyecto plantea, ya que no sólo propone una hipótesis sino también falsarla con la participación de los visitantes, que convertidos en sujetos y objetos del estudio pueden sacar sus propias conclusiones de lo que han presenciado. El enfoque que se dará al proyecto y a las conclusiones que puedan extraerse de la parte más performática es por necesidad estructuralista, conectando también de forma natural con un enfoque hasta cierto punto cientificista, al estar planteado el proyecto como una hipótesis y la propuesta de instalación como un experimento, si bien a diferencia del método científico no se busca reproducibilidad ni repetibilidad en los resultados.

El fenómeno de la emergencia permea todos los campos del saber y ha sido estudiado a lo largo de prácticamente toda la historia. En el presente trabajo se ha optado por constreñir el estudio a la emergencia de cariz biológico y las consecuencias sociológicas que ello comporta, haciendo especial énfasis en los autores que lo abordaron a principios del siglo XX. Esta limitación del estudio no se ha aplicado a los referentes artísticos dada la relativa escasez de obras que exploran estos conceptos, su cercanía relativa en el tiempo, y la variedad de enfoques interesantes que han implementado. Otra limitación que ha marcado de forma importante el enfoque de la obra es la monetaria: puesto que por su propia naturaleza la instalación requiere de un elevado número de módulos similares y el presupuesto total para cada uno de estos es por necesidad limitado. Incluso los motores más económicos disparan el presupuesto cuando necesitas decenas de ellos, con lo que se optó por prescindir de cualquier tipo de movimiento en la instalación. Lo que en principio parecía un *hándicap* animó a investigar otro tipo de sistemas en los que el movimiento no fuera un mediador de la interacción. El proyecto finalmente tomó fuerte inspiración de dos de estos ecosistemas: las colonias de luciérnagas norteamericanas que titilan en sincronía (Copeland & Moiseff, 2004) y el relativamente reciente descubrimiento de que los árboles se comunican entre ellos a través de redes de micorrizas (una

relación simbiótica entre árboles y hongos), alertándose entre ellos de los peligros y enviándose recursos vitales e información (Gorzelak et al., 2015).

Objetivos

El objetivo principal del proyecto *Emergencia* es la ideación y diseño de una instalación artística que ponga en valor y acerque al espacio expositivo los fenómenos de emergencia observados en sociedades animales, estableciendo para ello un marco de estudio de dichos fenómenos, así como de los acercamientos previos que se han hecho desde la práctica artística, política y sociológica.

Se deja para un futuro la implementación de la obra tal y como se plantea en el presente trabajo, siendo probada no obstante la viabilidad del mismo.

Se plantean como objetivos específicos para asegurar la consecución del objetivo antes mencionado:

- Analizar las características principales de la noción de emergencia en biología para establecer procesos similares en el devenir de la instalación.
- Contextualizar el proyecto *Emergencia* en el marco de las prácticas artísticas afines.
- Desarrollar debates en torno al discurso apoyo mutuo-individualismo

Se trabaja con la hipótesis de que la instalación actuará y será apreciada como una entidad única, pese a que las interacciones serán por necesidad y diseño módulo-módulo o módulo-humano.

Metodología

La metodología empleada es natural a la práctica de las artes visuales, y cabalga entre metodologías propias del ámbito científico, del ámbito práctico-empírico y del ámbito social.

La tesis nace empíricamente del análisis de múltiples teorías sociales y económicas que atañen al cómo se organizan las sociedades humanas, pero está claramente influida por los conocimientos previos en física, que muestra muchos comportamientos emergentes (física estadística, termodinámica, física cuántica, modelos físicos para explicar fenómenos sociales, etc) así como conocimientos tangenciales en biología (que también tiene tendencia a mostrar comportamientos emergentes) e informática, donde algoritmos como BOIDS (Reynolds, 1987) o el juego de la vida de Conway (Gardner, 1970) son ampliamente conocidos.

Ante tantos indicadores de un patrón multidisciplinar, se decide plantear una doble exploración: teórica y práctico-artístico-científica, que se retroalimenten. El estudio teórico dota de marco conceptual y base teórica a la práctica artística, que a su vez puede falsar el estudio teórico realizado. No se pretende así pues ilustrar con la práctica artística la hipótesis, ni se plantea dicha hipótesis como excusa para la realización de la práctica artística, sino que se desarrollan en paralelo nutriéndose mutuamente e influyendo en las direcciones que ambas toman.

De todos los ángulos con los que se puede afrontar el análisis teórico se decide limitar la investigación a los trabajos teóricos y prácticos que estudian la emergencia desde un punto de vista social (incluyendo en el aspecto social a sociedades no humanas), así como teoría básica anarquista, necesaria al afrontar un trabajo de esta tipología, todo ello dentro de los siglos XIX y XX.

La exploración artística es dirigida y modelada a imagen de sociedades animales, para ello se realiza una criba bibliográfica de la literatura científica que cubre los fenómenos emergentes observados en dichas sociedades, así como

de los papers publicados por artistas que han investigado este mismo fenómeno a lo largo de las tres últimas décadas.

Respecto al proceso de producción de la obra, se opta por una estrategia de prototipado rápido para el diseño de los diferentes módulos interactivos y de simulado por ordenador para el ensayo de modelos relacionales, lo que permite descartar vías fallidas y orientar la investigación hacia las más fructíferas.

2. Contexto

2.1. Emergencia, complejidad y caos

Cualquier sistema puede describirse de forma generalizada como un conjunto de subsistemas y las relaciones entre ellos. De hecho, esta capacidad de subdividir cualquier sistema, problema o cuestión en partes más sencillas es una de las razones del éxito de la ciencia tal y como la conocemos, ya que ha permitido adecuar y enfocar la investigación a objetivos asumibles por las herramientas disponibles en cada campo. Ahora bien, esta lectura de la realidad (todo sistema es sus partes más las interrelaciones entre ellas) es bidireccional: dado un conjunto de elementos o sistemas, si se dan interrelaciones entre ellos existe la posibilidad de que surja un súper-sistema de mayor complejidad y que no puede ser explicado simplemente por los elementos que lo conforman. Este fenómeno es el que conocemos como emergencia.

Por supuesto, esta es una definición un tanto vaga y no caracteriza el comportamiento del sistema en cuestión. Se nos pueden plantear múltiples preguntas: ¿qué tipo de interrelaciones son necesarias?, ¿qué entendemos por interrelación en cualquier caso?, ¿es el orden un requisito o una consecuencia de la emergencia?, ¿aplica esto a cualquier tipo de sistema?

Para dar mejor respuesta a estas preguntas conviene considerar la emergencia no tanto como un fenómeno en sí mismo (que lo es) sino como la frontera, o puente si se quiere, entre el caos y la complejidad. Conviene puntualizar no obstante que no es el único puente que existe entre ambos estados. Si la emergencia se puede entender como el puente espontáneo y auto-construido, la jerarquía se puede entender como el puente artificial y dirigido para modelar complejidad donde antes solo había caos. Esto da lugar a confusiones bastante comunes, como la contraposición jerarquía-caos, y la asunción de que sin la primera se obtiene inevitablemente la segunda. Las implicaciones sociales que ha tenido este tipo de pensamiento han sido muy profundas, y sistemas políticos

enteros se han desarrollado en torno a ambos modos de entender la auto-organización, pero esto lo analizaremos más adelante.

Por último, conviene tener presente que la emergencia no es frontera únicamente de la dualidad orden-caos, sino que también lo es de los diferentes niveles de explicación cuando hablamos de conocimiento.

2.2. En el borde del caos

El siglo XIX fue la cumbre del conocimiento. La física estaba acabada, las leyes de la mecánica que describían cómo interaccionan los cuerpos completamente formalizadas, y los métodos para calcular y predecir el futuro de cualquier sistema del que se conociese la información necesaria, asentados. Y aunque otras ramas de la ciencia estaban en auge -como por ejemplo, la biología-, en el fondo éstas eran redundantes. Al fin y al cabo, los animales se pueden describir mediante biología, ésta mediante bioquímica, que es una especialización de la química, que no deja de ser física aplicada. Y la física estaba resuelta. Como dijo Laplace en 1814:

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes. (Laplace, 1814)

Tres conceptos destrozarían el sueño determinista: el caos, el azar, y la emergencia.

2.3. Definiciones. La emergencia en las ciencias puras.

Cabe precisar que un sistema caótico no es necesariamente un sistema azaroso, si bien suelen ir relacionados, con lo que no es extraño asociarlos. No obstante, sobre el papel son dos cosas muy distintas.

Caos es el tipo de comportamiento que se da en sistemas dinámicos altamente sensibles a sus condiciones iniciales. Es posible calcular con exactitud el comportamiento del sistema, pero a poco que cambie el punto del que empieza, el resultado será radicalmente distinto. Edward Lorenz, uno de los padres de la teoría del caos lo sumarizó en una frase:

> Chaos: When the present determines the future, but the approximate present does not approximately determine the future.(Jones, s. f.)

Puesto que cualquier cálculo que hagamos tendrá siempre precisión limitada, nunca podremos calcular el futuro de ningún sistema que sea caótico, porque nunca podremos acercarnos lo suficiente a sus condiciones iniciales. Y no hace falta irse muy lejos para encontrar sistemas caóticos: si bien las ecuaciones de Newton describían con admirable precisión las trayectorias de dos planetas interactuando entre ellos, fracasaban espectacularmente en cuanto se introducía un tercero. Y no solo las órbitas planetarias son caóticas, también lo es el clima, los movimientos de aire o de agua, o incluso los mercados financieros.

Azar es aleatoriedad. La más absoluta impredicibilidad. Aunque en la práctica los procesos caóticos pueden parecer azarosos, en teoría si se tuviese una precisión infinita se podrían predecir. Esto no ocurre con el azar. Se creía que era imposible, un truco matemático. Científicos de la talla de Einstein se resistieron con uñas y dientes a la idea de que el azar existía, y sin embargo la mecánica cuántica asentó definitivamente la existencia del azar en la física. No podemos predecir el futuro, porque depende de un lanzamiento de dados.

Roto el sueño del determinismo cabía preguntarse qué podía salvarse. Y se salvaron cosas. Con algunos trucos matemáticos muy ingeniosos se hizo un

importante salto cualitativo: no puedo calcular lo que hacen *tres* planetas/cuerpos/partículas, pero sí lo que hacen *infinitos* si me aprovecho de la estadística. No puedo determinar la posición de *un* fotón, pero sí lo que van a hacer *millones* de ellos. La estadística no te permite estudiar lo individual, no puedes saber si la próxima tirada será cara o cruz, pero sí que si tiras millones de monedas, más o menos la mitad habrán caído de un lado y la otra mitad del otro. Más o menos. Esto es un salto cualitativo nada trivial. Surgen conceptos que solo tienen sentido para lo colectivo, y que no vienen explicados por las partes individuales.

Este mismo salto cualitativo lo dieron los psicólogos cuando se hizo patente que no se podía explicar la psicología a partir de los fenómenos neurológicos que se dan en el cerebro. Acuñaron el concepto de los niveles de explicación: una misma realidad puede ser entendida, estudiada y explicada a diferentes niveles de abstracción, ofreciendo cada uno de ellos una explicación cualitativamente diferente al resto, y que no podría ser obtenida en los otros niveles de concreción.

La química no es física aplicada. Es física más emergencia. La biología no es química aplicada. Es química más emergencia. Y así hasta el infinito.

El reduccionismo científico ha dado sus frutos, pero en muchas ocasiones ha obviado que los fenómenos estudiados son holísticos, y no se pueden reducir únicamente a sus partes constituyentes.

Este tipo de salto que define una frontera se encuentra en todo tipo de sistemas, y es en las fronteras donde a menudo ocurren las cosas de interés. A nadie le interesa lo que ocurre con el agua cuando está a 17º, pero es por todos conocido que a 0º y a 100º ocurren cosas interesantes. Al aumentar la temperatura añadimos energía al sistema, y cuando alcanzamos esos umbrales, cambia radicalmente la forma de organizarse a nivel molecular de dicho elemento. Quiero incidir en que las moléculas de agua son las mismas: son las relaciones entre ellas las que se ven afectadas, teniendo eso una repercusión en el colectivo.

No sería yo la primera persona en trazar paralelismos entre este tipo de comportamientos colectivos y los comportamientos colectivos sociales – ya Isaac Asimov describió esta idea profusamente en su serie de novelas *La Fundación* (Asimov, 1966)–, sin embargo no lo haré. Los modelos físicos son interesantes, pero no aplican con exactitud a los fenómenos sociales. Lo que es indudable es que es en la frontera de la abstracción donde en ocasiones ocurren los fenómenos más interesantes, ya sea en ciencias exactas o en un grupo de personas intentado progresar en la vida, y que un análisis de las relaciones entre elementos y sus implicaciones en el colectivo es un análisis que vale la pena realizar.

2.4. Emergencia natural

> There is grandeur in this view of life, with its several powers, having been originally breathed by the Creator into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been and are being evolved.
>
> (Darwin et al., 2013)

Ya hemos comentado que es en la frontera entre el caos y el orden donde ocurren los fenómenos más interesantes gracias a las interacciones entre los elementos. Uno de los fenómenos más preciosos que se pueden describir mediante este mecanismo es precisamente la vida, y es la selección natural uno de los mecanismos de interacción más significativos que conocemos, si acaso porque gracias a él podemos describir esto. Vale la pena detenernos a explicar exactamente qué es y qué no es selección natural, ya que existen múltiples confusiones al respecto. La selección natural no entiende de intencionalidad. Las especies *no quieren sobrevivir.* El instinto de supervivencia es prácticamente universal, y parece un mecanismo obvio, pero no deja de ser una característica evolutiva que vemos porque es favorable a la supervivencia. ¿De la especie? No, *de la propia característica*. Las especies, los seres vivos, son desde este

punto de vista *los vehículos*. Por mecanismos azarosos se introduce variabilidad en las características de una especie; si una de esas características ofrece algún tipo de ventaja en el contexto del entorno a esa especie, esa característica se va haciendo prevalente por medio de la descendencia –quienes no tengan dicha ventaja morirán más fácilmente en comparación, o se reproducirán menos – . Esto nos lleva a otros dos puntos donde suele haber confusión: las especies *no se intentan adaptar al medio*: es la presión ambiental la que propicia que ciertas adaptaciones prosperen o no. Pero sobre todo, y a donde quiero llegar: *nadie dijo nunca que sobrevive el más fuerte*.

Al menos nadie con una reputación sólida en biología evolutiva. De hecho esa frase fue acuñada por Herbert Spencer, padre del conocido como Darwinismo Social, una visión centrada en el individuo y en la prevalencia del más fuerte de las teorías de Darwin aplicadas al ámbito de lo social, tratando así de justificar no sólo el ultraliberalismo como si de una ley natural se tratase, si no el racismo sistémico del momento. No es de extrañar que su corpus teórico fuera base del ala teórica del nazismo.

Cabe destacar que el Darwinismo Social no tiene ninguna base teórica, no siendo muy difícil encontrar centenares de contra-ejemplos en la naturaleza. Se puede observar entre especies (relaciones simbióticas) pero también, y sobre todo, en comunidades dentro de una misma especie. En efecto, la creación de sociedades parece ser un gran factor evolutivo que encontramos a lo largo y ancho del mapa filogenético en menor o mayor medida, con ejemplos que abarcan desde el sacrificio del individuo en favor de un bien grupal hasta los cuidados cruzados.

Piotr Kropotkin, biólogo ruso nacido en 1842 notaría esto y recogería múltiples ejemplos de lo que llamaría cuidados mutuos en la naturaleza, probando que el interés individual supremo no suele ser lo que guía y hace evolucionar a las sociedades animales, no en todos los casos al menos. Kropotkin extendería este pensamiento a las sociedades humanas, y construiría todo un corpus teórico en base a eso en su obra *El Apoyo Mutuo* (Kropotkin et al., 2016), siendo considerado uno de los padres teóricos del anarquismo moderno.

2.5. Emergencia social

La propuesta de Kropotkin choca frontalmente con la concepción del mundo moderno: si bien el Darwinismo Social quedó desterrado como teoría biológica, su impacto en los mercados perduró. El modelo de gestión de lo público neoliberal es por lo general poco discutido, e incluso cuando encuentra oposición suele ser hacia el *qué* y *cómo* se gestiona lo que hacía el modelo en sí[1]. Lo cierto es que, al menos en occidente, el pensamiento de mejora de lo social a través de la búsqueda de la mejora individual es prevalente, y la creencia de que los mercados se autoregulan satisfactoriamente, muy extendida.

La economía bajo régimen de mercado es un modelo caótico (usando la misma definición de caótico antes mencionada), pero caótico no implica no modelable. Me permito insistir: caótico no significa aleatorio, y aunque los modelos que podamos hacer no permiten predecir el futuro con certeza aceptable, sí permiten extraer patrones e información. No pretendemos aquí presentar un modelo de un sistema de libremercado, y tampoco consideramos que sea necesario para sostener una crítica fundamentada al capitalismo como esqueleto de la sociedad moderna: otros han hecho ya esa tarea, y de forma más fundamentada de lo que podamos hacer aquí (lucha de clases, la fantasía del crecimiento infinito, la no libertad de negociación, las desigualdades de información, la falacia del actor racional) pero si algo nos proporciona el conocimiento empírico es que:

- El capitalismo neoliberal conduce inexorablemente a la acumulación de capital, con el consecuente crecimiento de la desigualdad económica en la sociedad

- Dicho sistema conduce a crisis económicas cíclicas – hecho reconocido incluso por los más importantes teóricos de la escuela austríaca

Es por tanto cabal pensar en alternativas a un sistema que plantea los problemas descritos y que no está sustentado por la justificación que se le suele

1 La realidad, como todo, es más compleja, y prácticamente todos los países operan en mayor o menor medida bajo un modelo mixto público-privado

adjudicar. La propuesta de Kropotkin es sin duda interesante, pero interesante no implica viable. ¿Es posible concebir un modelo de sociedad basado en apoyo mutuo y la horizontalidad en lugar de en la verticalidad y optimización del beneficio propio? Lo cierto es que ese tipo de comportamiento emergente existe y se da en varios ámbitos y niveles de aplicación.

En primer lugar, cuando la ambición del beneficio (que no supervivencia) propio desaparece suelen surgir rápidamente redes de apoyo: esto se ve en todas y cada una de las catástrofes una vez el pánico inicial desaparece. A la hora de reconstruir tendemos puentes, y somos capaces de anteponer el bien social al beneficio particular.

No hace falta irse a situaciones tan dramáticas para observar el mismo tipo de fenómenos: cuando se da el marco de actuación adecuado la gente espontáneamente colabora. Esto es muy común de ver en el circuito artístico, especialmente en el ámbito de la improvisación. Se podría argumentar que en esas situaciones el arte actúa como vehículo mensajero, y que lo único que ocurre es una conversación entre individuos mediante un medio diferente. Tal vez. Pero también tenemos muestras como *r/Place* (2017), un experimento de Reddit en el que cada usuario podía alterar el color de un solo píxel en un lienzo común. Un píxel no permite establecer una comunicación significativa, pero millones de personas lograron establecer y colaborar de una forma que desde el individuo no tenía sentido.

Eso fue posible gracias a internet, otro de los grandes catalizadores de este tipo de fenómenos. El software libre y los proyectos abiertos son un perfecto ejemplo del tipo de emergencia de la que hablamos: colaborar en wikipedia en rara ocasión aporta un beneficio personal (se pierde un tiempo precioso en redactar algo que ya se conoce y por lo que no se va a percibir ningún tipo de remuneración económica, social o de prestigio), y sin embargo la autocoordinación de millones de personas ha producido una enciclopedia con una calidad similar o superior a la británica.

2.6. Aproximaciones a la emergencia desde el arte

Muchas veces, y desde muchas perspectivas se ha abordado la emergencia desde el arte. Ya he mencionado el proyecto r/Place (*The /r/place Atlas*, s. f.), pero no es sin duda el único. r/Place es, no obstante, bastante especial en su planteamiento. Es de las pocas obras en las que se plantea (literalmente) un lienzo, dejando que el resto del mundo lo rellene. No hay planteamiento, no hay dirección, tan solo un posible (¿emergerá algo de esta propuesta?) y un resultado.

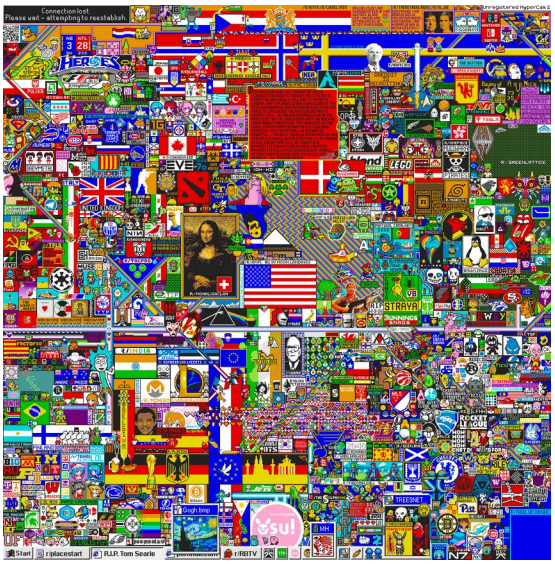


Figura 1: Versión final de r/Place

El resto de obras referenciadas sí cuentan con un discurso y un planteamiento claro, y han resuelto los retos que plantean las instalaciones emergentes de diversas maneras.

Simulaciones - arte generativo

Es el que conforma la forma más básica de emergencia. Dadas una serie de reglas, el ordenador simula la evolución del sistema. Tiene un trasfondo fuertemente matemático, y no es para menos: fue un matemático quien originó esta disciplina, el ya mencionado Conway con su *Juego de la Vida* (Gardner,

1970). La disciplina ha avanzado mucho desde entonces, con artistas que se mueven entre complejas simulaciones herederas directas del juego de la vida (Sebastian Lague, 2021) hasta modelos propios y sofisticados que apuntan más en una dirección estética/artística como la totalidad de la obra de *Inconvergent* (Hoff, 2016) o trabajos conceptuales como el álbum *Emergence* (Cooper, 2016)

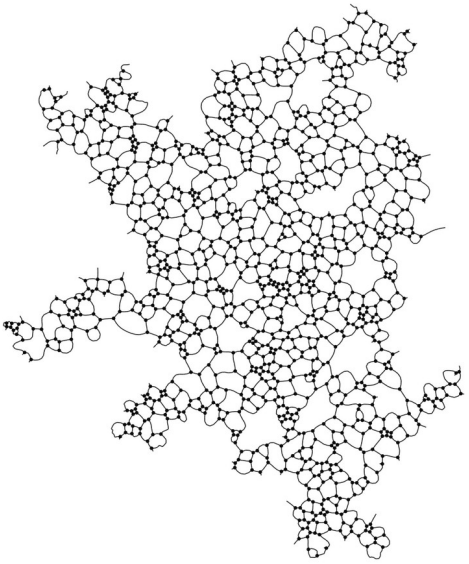


Figura 2: Differential Lattice. Inconvergent *(2015)*

Instalaciones – game of life

Algunos artistas han llevado al espacio expositivo el mismo tipo de piezas que Conway simulaba en el ordenador, siendo ejemplos destacados de ello *Synaptic Caguamas* (Lozano-Hemmer, 2004), una implementación directa del juego de la vida traída al espacio físico.

Una aproximación similar tomé con mi anterior obra *Escacs Temporalment Autónoms* (2020) en la que traía al espacio físico una interfaz para el juego de la vida basada en un tablero de ajedrez modificado, puenteando parte de la obra a una simulación por ordenador.

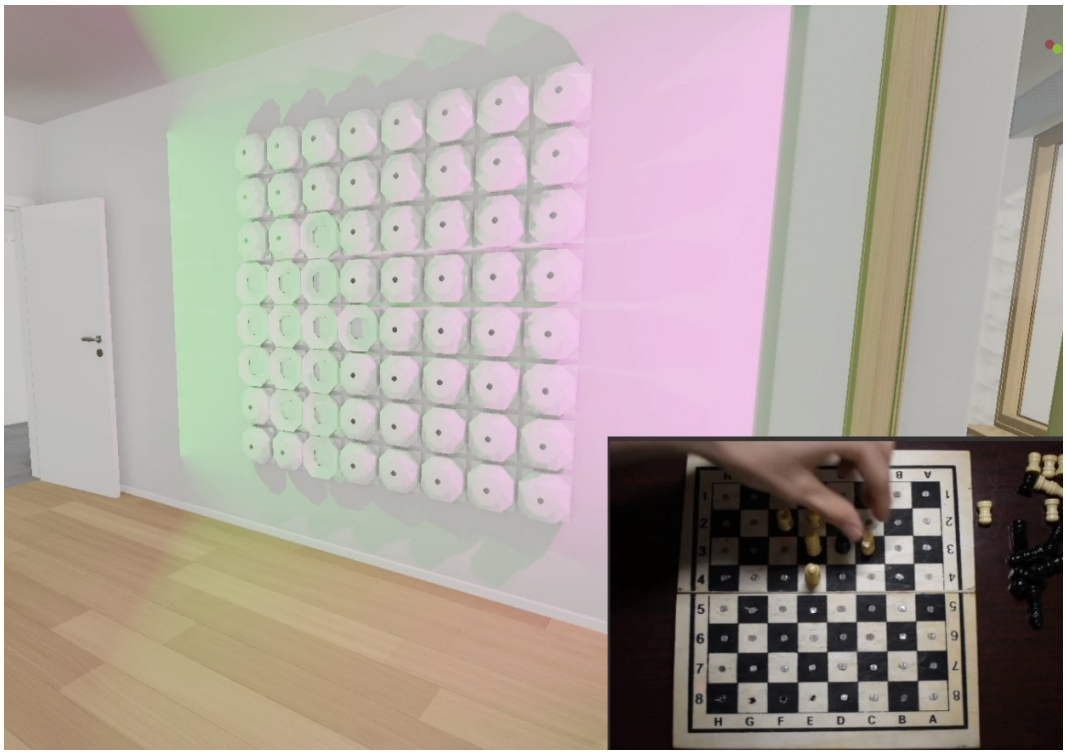


Figura 3: Escacs Temporalment Autònoms, (2020) pieza a caballo entre lo físico y lo virtual en la que un tablero de ajedrez actúa como entrada para un Juego de la Vida en tiempo real

Un proyecto más sencillo pero con la misma esencia es *Constellaction* (panGenerator, 2013), pieza conformada por decenas de pequeños módulos reactivos a la luz que da pie a patrones complejos cuando interaccionan entre sí y con el público.

Mucho más elaborada, pero basada en el mismo principio del juego de la vida es la instalación *Edge of Chaos* (Abramovic & Glynn, 2018), un complejo armatoste reactivo a la presencia de quienes lo visitan, pero con las mismas reglas básicas operando por debajo, o la instalación *Impulse* (Martin Messier, 2018), que plantea una visión cruda de lo que son las redes neuronales.

Simulaciones – swarm

De origen similar (simulaciones de ordenador en los 70) tenemos los códigos de tipo swarm, esto es, simulaciones de sociedades de organismos interactuando entre ellos. El primer ejemplo de esto es el código BOIDS (Reynolds, 1987), que simula bandadas de pájaros.

Figura 4: Life Writer (2006) Christa Sommerer & Laurent Mignonneau .

Una de las grandes inspiraciones conceptuales del presente trabajo, la obra casi íntegra de Christa Sommerer y Laurent Mignonneau (s. f.) está basada en gran medida en este tipo de algoritmos. Cuenta con centenares de artrópodos que se comportan como una colonia e interaccionan con el entorno a través de diversas instalaciones físicas, con un discurso muy potente en torno a la adaptabilidad al medio.

3. Planteamiento y desarrollo de la obra

3.1. Planteamiento del proyecto

Se propone así el desarrollo y producción de una obra en la que los fenómenos emergentes y sus implicaciones socioculturales se vean reflejados y formen parte de su diseño e implementación, situándola en la intersección entre la ciencia de ecosistemas, la sociología y el arte.

En el proceso de investigación previo al planteamiento de la obra se comprueba que el fenómeno de la emergencia se suele afrontar desde los siguientes ejes transversales:

- Simulación/fisicidad de las interacciones entre los elementos de la obra
- El grado de sociología que implica el sistema representado por la obra

En el primer eje distinguimos entre obras que implementan las interacciones de una forma física o que las simulan de forma centralizada, generalmente desde un ordenador. No apelamos a la fisicidad de la obra resultante: instalaciones tales como *Edge of Chaos* (Abramovic & Glynn, 2018) si bien son instalaciones físicas son controladas de forma centralizada por un ordenador.

Figura 5: Edge of Chaos (2018). Vasilija Abramovic &Ruairi Glynn.

En el segundo eje contrastamos la intencionalidad de la obra. La línea que separa la vida de lo inerte es difusa incluso para los biólogos, y esta no es una excepción, si bien es cierto que podemos distinguir a grandes rasgos los proyectos que se limitan a reproducir algoritmos que dan lugar a fenómenos colectivos, tales como *Game of Life* (Bailey, 2020), o los que emulan sociedades más avanzadas tales como las colonias de artrópodos de Christa Sommerer y Laurent Mignonneau (s. f.).

Hay más ejes que no nos paramos a analizar, como la interactividad de la obra, ya que la única diferencia a nivel conceptual es la integración del espectador en la obra o no: dadas sus características ninguna de las obras analizadas *necesitaba* de dicha interacción.

Al situar las obras más referenciadas en una tabla notamos algo interesante:

	físicas	**simuladas**
algorítimico	Edge of Chaos	Game of Life
	Constellaction	Arte generativo
	Impulse	
	Synaptic Caguamas	
life-alike	---	Los artrópodos de Christa Sommerer y Laurent Mignonneau

Emergencia se sitúa en un nicho poco explotado: el de las obras de ejecución física y vínculos físicos pero que se alejan de comportamientos meramente algorítmicos.

Nos distanciamos así de trabajos propios previos, tales como *Escacs Temporalment Autònoms* (2020), en los que también se estudiaba el fenómeno de la emergencia y se confrontaba con un discurso social, pero desde un punto de vista totalmente reactivo y con un enfoque centralizado, utilizando un tablero de ajedrez como interfaz de un juego de la vida tradicional.

3.1.1. El diseño basado en restricciones

> “Au fond, je me donne des règles pour être totalment libre”.
>
> (George Perec en L’ OuLiPo, 2012-2013, p.3)

La selección natural no es una historia de adaptación, sino de supervivencia. En un entorno en el que aparecen unas limitaciones para el desarrollo, solo los individuos preparados por puro azar para sobreponerse a ellas son capaces de otorgar a su descendencia esa ventaja fortuita.

Las soluciones a cualquier sistema de ecuaciones de un modelo físico no significan nada hasta que se aplican dos elementos: unas condiciones iniciales, y las limitaciones en las que el sistema se desarrolla.

Toda obra de arte se desarrolla en un contexto, y es este contexto el que determina, a través de las limitaciones que impone, cómo dicha obra de arte se produce.

Contra las limitaciones se puede luchar, se pueden esquivar o incluso en ocasiones, doblegar. Las limitaciones se pueden aceptar, trabajar con ellas, o incluso trabajar sobre ellas. Las limitaciones se pueden incluso autoimponer en un ejercicio creativo.

Emergencia es un proyecto que cuenta así mismo con restricciones -tanto exógenas como endógenas- y son estas restricciones las que han modelado y dirigido en gran medida la forma final que ha tomado.

Podemos clasificar las restricciones a que se ha enfrentado el proyecto en tres grandes categorías: de presupuesto, de tiempo y de complejidad.

Restricciones de presupuesto:

Emergencia es un proyecto de escala. Por necesidad requiere de múltiples elementos interaccionando entre sí, y en este caso múltiples no puede ser

menos de 20-30 elementos. Idealmente se pueden plantear instalaciones de hasta 200 elementos, o incluso más. La planificación de escala requiere tener en cuenta que dicha escala aplica también a los márgenes de error con que se cuenta; un sobrecoste de tan solo 3€ por módulo se traduce en un sobrecoste de unos 100€ en el proyecto. Para una obra no financiada como la que aquí se presenta es un factor a tener muy en cuenta dada la limitación de presupuesto: si no se quiere gastar más de ~120€ en la obra. el coste por pieza no debiera exceder los 4-5€ en ningún caso. Teniendo en cuenta que el microcontrolador de cada uno de los elementos ya cuesta aproximadamente 3€, contamos con unos 2€ máximo para diseñar el resto del elemento.

Eso deja fuera del diseño opciones como motores, sensores especializados, o actuadores varios. Queda por tanto descartado el movimiento como elemento de interacción o reacción, limitándonos a lo que podamos hacer mediante señales visuales, sonoras o eléctricas.

Restricciones de tiempo:

Los mismos problemas de escala que afectan al presupuesto afectan al tiempo requerido para desarrollar la obra. Al margen de la inespecificidad que implica trabajar sobre fenómenos emergentes que por definición no van determinados por los parámetros de configuración del proyecto, cualquiera que sea el tiempo requerido para ensamblar, programar, y depurar uno de los módulos habrá de ser multiplicado por 30 a la hora de montar la pieza final. Eso implica adoptar diversas estrategias de desarrollo, como son la inversión de tiempo en simulaciones y maquetaciones 3D que de otra forma serían demasiado costosas. En concreto las simulaciones por ordenador del comportamiento final de la pieza han permitido iterar, descartar y orientar el código final sin tener que pasar por el proceso de programar uno a uno 30 módulos, con las complicaciones logísticas que eso conlleva.

Restricciones en cuanto a complejidad:

La complejidad del proyecto viene determinada en gran medida por los dos apartados anteriormente mencionados, con el añadido que cualquier modelado de un sistema en que los elementos hablan entre sí es un modelado de una red

de información. Existen disciplinas enteras que versan sobre teoría de redes y diseño de protocolos de comunicación, y en gran medida los problemas que se plantean han sido resueltos con anterioridad, pero conviene no perder de vista que, siendo que el proyecto está construido desde primeros principios, cualquier protocolo de comunicación que se implemente tendrá que hacerse desde cero. En el caso que nos ocupa se plantean dos supuestos y un problema.

Los supuestos son la posibilidad de un protocolo de comunicación digital, con información codificada en base a dicho protocolo, o un protocolo de comunicación analógica basado en señales sencillas. El primer supuesto, *a priori* más atractivo dadas las posibilidades que permite y para nada descabellado siendo que todo el proyecto está basado en microcontroladores preparados para ello, implica el diseño de un protocolo de comunicación para nada trivial, que requiere una sincronización perfecta entre dispositivos, una programación nada sencilla, y muy poca resiliencia ante el problema antes mencionado: pérdida de información. Se plantea entonces la segunda opción: comunicación analógica basada en enviar señales al entorno de forma ciega, y recibir señales sin discernir de dónde vienen o cuántas llegan. Un modelo unidireccional en el que no se establece realmente diálogo entre dispositivos, pero mucho más resiliente y sencillo de implementar.

3.1.2. Sistemas orgánicos con similares restricciones

De entre todo el abanico de fenómenos emergentes se decide tomar inspiración de aquellos que atañen a sistemas orgánicos dado el carácter de la obra. Se realiza una investigación para determinar cómo diversas especies han solucionado los *hándicaps* mencionados, así como sus mecanismos de interacción.

El primer candidato considerado, en parte por inspiración en la obra *Life Writer (2006)* de Christa Sommerer y Laurent Mignonneau (s. f.), las colonias de

artrópodos, presenta comportamientos gregarios muy interesantes y establece el germen de lo que será la obra final; dispositivos pequeños inspirados en pequeños insectos que establecen estructuras similares a colonias. No obstante presentan un importante problema: median la comunicación a través del movimiento.

Se buscan así sistemas en los que el movimiento no sea clave en el proceso emergente.

El de las luciérnagas es uno de estos ecosistemas; si bien son insectos capaces de volar, el fenómeno que nos interesa es puramente estacionario. Hablamos de la sincronía espontánea que manifiestan algunas especies de luciérnagas norteamericanas (Copeland & Moiseff, 2004), fenómeno en el que se inspiran obras tales como *Luci* (Berenguer, 2007). En un fenómeno asimilable a un modelo de oscilaciones acopladas forzadas, cada una de las luciérnagas tiene un período de centelleo diferente, pero al percibir el centelleo de luciérnagas próximas modifican temporalmente el suyo propio. En este punto se plantea imitar este comportamiento, dotando a los dispositivos de emisores y sensores de luz. Nos topamos no obstante con un impedimento, tanto las luces LED como los sensores de luz son altamente direccionales, lo que imposibilita una buena implementación de esta propuesta como comprobamos rápidamente en nuestras simulaciones. No obstante el prototipo hereda los LEDs como dispositivo de comunicación hacia el exterior, si bien no como canal de comunicación intra-colonia.

Se propone entonces comunicar los dispositivos mediante cables o buses de datos. Hay por lo menos dos ecosistemas con una configuración similar a la que se plantea: la simbiosis micorriza y los sistemas neuronales.

La simbiosis micorriza

Las micorrizas son una simbiosis que se da entre hongos y las raíces de los árboles. Se conocen desde hace más de medio siglo (Gorzelak et al., 2015; Björkman, 1970; Francis & Read, 1984; Newman, 1988; Reid & Woods, 1969), si bien ha sido en las últimas décadas cuando se ha empezado a estudiar las redes de micorrizas (Simard et al., 1997; Helgason & Fitter, 2005). Concretamente se ha descubierto no sólo la fuerte prevalencia de árboles micorrizados sino el papel que la red resultante desempeña en la supervivencia de los bosques en que se encuentran. Dicha red, que comunica los árboles a los que pertenece sirve de intermediaria de señales y recursos, alertando al resto de miembros de la red de amenazas, o transportando nutrientes en caso de necesidad. Son redes multi-especie (tanto en el lado de los árboles como en el de los hongos), descentralizadas pero no plenamente horizontales (no todos los árboles están conectados al mismo número de nodos, adquiriendo importancia en la red conforme envejecen, llegando algunos a ser piezas claves dentro del bosque).

Si bien los elementos en *Emergencia* son similares entre sí y por tanto no hay roles diferenciados o desequilibrios en ese sentido, sin embargo sí se decide implementar la red densa de comunicaciones y emular el comportamiento ante amenazas, así como la red compartida de alimentación (en este caso, eléctrica).

Los sistemas neuronales

Presentes como sistema nervioso en gran parte de los animales, los sistemas neuronales han permitido la evolución de varios tipos de inteligencia, e implementaciones análogas en el campo de la informática han mostrado su capacidad y flexibilidad de adaptarse a problemas siendo una buena base para la inteligencia artificial.

Los sistemas nerviosos son altamente complejos y una descripción fidedigna de su funcionamiento escapa a la intención del presente trabajo, pero pueden ser simplificados como un conjunto de elementos (neuronas) y conexiones, así como un estado (excitado/sin excitar) de dichos elementos y la capacidad de mandar señales a los otros elementos conectados al propio. En general se considera que para que se produzca la activación de una neurona es necesario que se supere el conocido como "umbral de activación", esto es, que reciba simultáneamente o en un breve lapso de tiempo varias señales de las neuronas colindantes. Las conexiones neuronales se ven fortalecidas o debilitadas con el uso o la falta de él, lo que afecta directamente a la estructura de la red.

Cuando hablamos de redes neuronales informáticas nos basamos en el mismo mecanismo, otorgando a las conexiones diferentes pesos que ponderan la señal enviada, siendo la calibración automática de estos pesos para optimizar el funcionamiento de la red el proceso conocido como "aprendizaje"

3.2. Prototipado iterativo

Partiendo de la idea inicial (sociedad de elementos electrónicos que se comunican entre sí, de forma visible, y cuyos vínculos pueden ser alterados por la presencia e interacción de los visitantes de la obra) se plantean una serie de prototipos, cada uno fruto del anterior y las limitaciones encontradas.

3.2.1. Prototipo 1: torretas láser

Se plantean pequeñas torretas estacionarias del tamaño de una lata. Como inputs tendrían sensores de luz, como outputs diodos láser, y su forma de comunicarse sería a través de su propia rotación. La hipótesis era que con determinadas reglas de interacción se establecerían redes semiestacionarias y comportamientos colectivos.

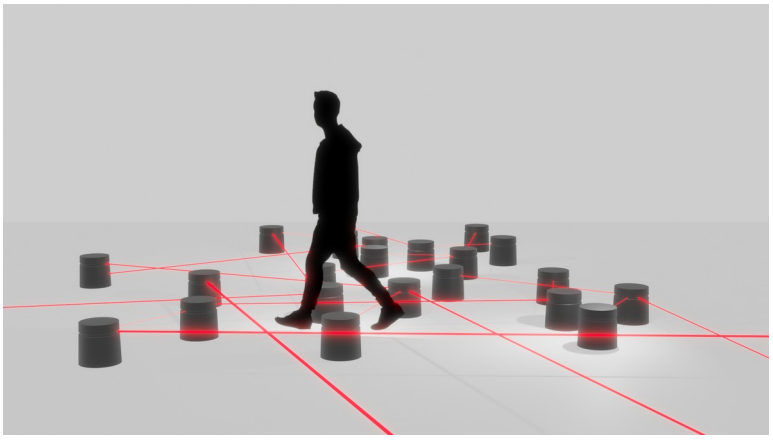
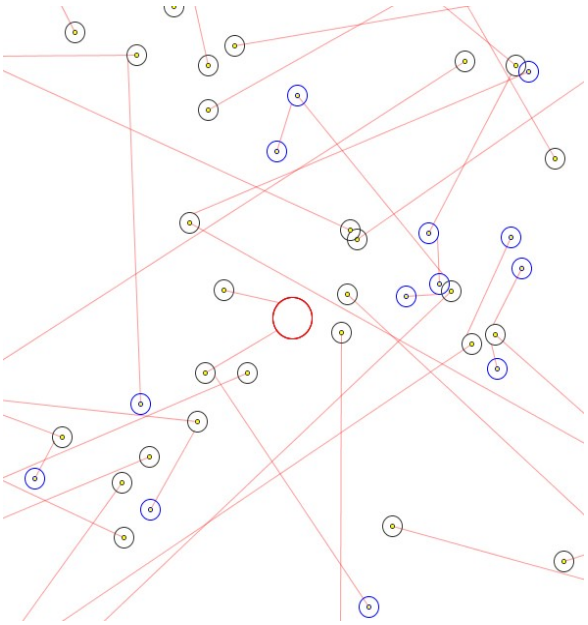


Figura 6: Mockup del prototipo 1



Se implementa un prototipo en *Processing* que muestra que el modelo no funciona: la combinación de un tipo de comunicación unidireccional con un mecanismo de comunicación de respuesta tan lenta (la rotación de la torreta) y establecimiento de canal tan fortuito (que se alineen el emisor láser con el receptor de otra torreta) provoca que si se establece algún tipo de patrón, este sea ininteligible.

*Figura 7: S*imulaciones del primer prototipo, vista cenital

Implementar este prototipo sirvió también para aprender programación orientada a objetos en Processing y técnicas de *raycasting* que serían usadas posteriormente. El *raycasting* es una técnica por la que se puede "lanzar" un rayo virtual desde un punto de la pantalla hasta que se encuentra con un obstáculo, obteniendo información tal como la distancia recorrida, la dirección que tiene, etc.

La implementación contempla dos tipos de superficies detectables por uno de estos rayos: círculos y líneas. Cada "torreta" consiste en uno de estos círculos y uno de los rayos con los que realizamos *raycasting*. Cada torreta modula la velocidad de su propio rayo en función de si su círculo detecta estar siendo alcanzado por un rayo diferente. La escena consta de un número de "torretas" aleatoriamente distribuidas y con velocidades de giro también aleatorias. También se implementó un obstáculo manejable con el cursor para simular la interacción de personas con la obra. Se probaron diferentes tipos de modulaciones de la velocidad: desde cambios de giro, transferencia de velocidad, ralentizamiento, etc. sin éxito.

3.2.2. Prototipo 2: luciérnagas

Sin abandonar la idea de comunicación en base a la luz se descarta el uso del movimiento como vehículo del canal de comunicación. Se encuentra en las sociedades de luciérnagas norteamericanas mencionadas anteriormente una fuente de inspiración, tanto en el aspecto conceptual, en el modelo de interacción, y en la ejecución artística de la obra.

El modelo planteado es sencillo: cada "luciérnaga" dotada de un emisor y un receptor de luz tiene una frecuencia de parpadeo propia, frecuencia que es alterada cuando la luz receptada por el sensor sobrepasa cierto umbral (es decir, es modificada por el parpadeo del resto de "luciérnagas" adyacentes).

La simulación de este modelo es una implementación mucho más sencilla, con un único tipo de objeto: el círculo. El círculo tiene conocimiento de todos los otros círculos y actúa en consecuencia del estado de estos. La simulación tiene parámetros para los umbrales de sensibilidad, el radio de acción, el número de elementos, etc. Los elementos se pueden mover y activar en tiempo real.

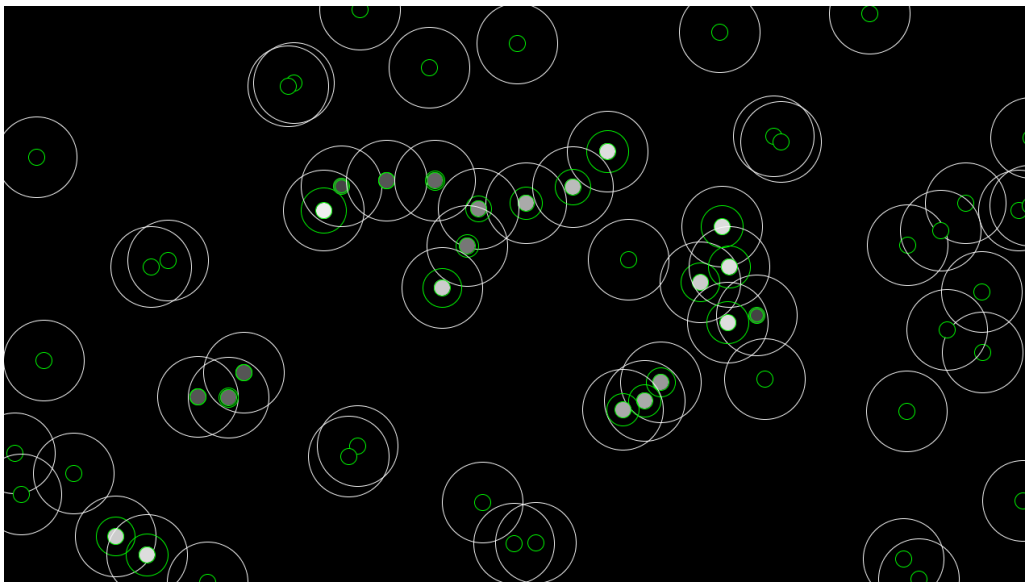


Figura 8: Simulación de elementos que se comunican mediante señales de luz visibles

Se desarrolló con el microcontrolador Arduino un framework de comunicaciones basadas en luz, que permitía mandar mensajes sencillos y procesarlos de forma asíncrona, logrando una cierta sofisticación desde el punto de vista del software.

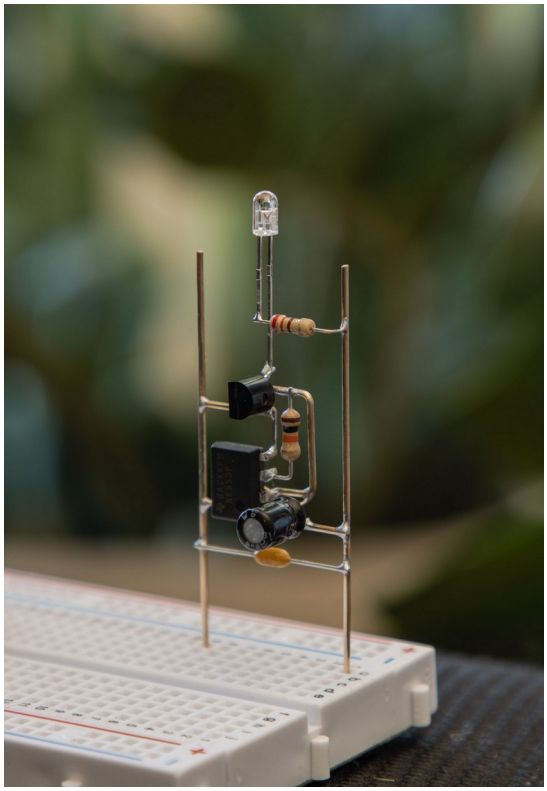


Figura 9: Escultura electrónica "free form" de Mohit Bhoite

Siendo que los artrópodos son estructuralmente concisos, disponiendo generalmente de un exoesqueleto sin ningún tipo de tejido muscular prominente, vellosidad, plumas o similares, se opta por inspirar la estética de la obra en el mismo tipo de "simplicidad desnuda". El "free form" es una corriente estética, observable en la obra de, por ejemplo, Mohit Bhoite (Hackaday, 2019). El "free form" entiende como material escultórico los propios elementos eléctricos que conforman la obra, actuando los cables, leds, y demás componentes de la circuitería como elementos estructurales de la pieza.

Este planteamiento presenta dos problemas: uno conceptual y uno práctico. Conceptualmente se pierde la interacción espectador-obra, ya que si bien es

posible intervenir las comunicaciones (son ópticas al fin y al cabo) el impacto es lo bastante pequeño como para que no se altere perceptualmente el comportamiento de la obra. Por otra parte nos encontramos con una severa limitación técnica: tanto los diodos LEDs como los fotorreceptores disponibles (LDRs) son altamente direccionales. Esto, que a priori puede no parecer un problema, implica la imposibilidad de establecer canales de comunicación en los que un receptor recibe información de más de un emisor, imposibilitando así la creación de una red de interacciones.3.2.3. Prototipo 3: luciérnagas arbóreas

Se decide adaptar la pieza y hacer que, manteniendo el modelo y la expresión basada en luz, las señales se manden mediante un entramado cableado, haciendo así referencia a las simbiosis micorriza mencionadas en el apartado 3.1.2.

Aprovechando la naturaleza "free form" de la pieza y su inspiración en las redes micorrizas se plantea que el exoesqueleto sea un sensor capacitivo que al ser tocado por los visitantes lance una señal de alarma que se distribuya rápidamente por toda la pieza.

Al realizar modelos físicos de estos elementos quedaría patente no obstante que la pieza no invita a la interacción con ella, por lo que se sustituiría este exoesqueleto capacitivo por una antena sensible a perturbaciones en el campo electromagnético circundante, dotando a estas "luciérnagas" de un sentido del entorno y la capacidad de detectar interacciones a distancia.

3.3. Proceso realización prototipo 3

3.3.1. API: entradas/salidas requeridas por módulo

Existen dos aproximaciones a la transferencia de información entre módulos: codificada o sin codificar. Si se codifican los mensajes se puede enviar y/o recibir más de un tipo de mensaje por canal, a expensas de una mayor complejidad de

dichos mensajes y la necesidad de un protocolo de comunicación. Los protocolos de comunicación no son triviales de implementar, y enfrentan dos problemas:

- El reloj: hace falta una sincronía milimétrica entre emisor y receptor. En sistemas centralizados se puede solucionar hasta cierto punto con un reloj central, pero en sistemas descentralizados como el que nos atañe el problema toma características dignas de tesis doctoral en ciencias de la computación y de sistemas.

- Las colisiones: Puesto que en un mismo canal esperamos recibir mensajes de más de un emisor es posible que dos mensajes lleguen a la vez, corrompiéndose y haciendo ininteligible la información recibida. Existen soluciones y protocolos establecidos para solucionar este problema, pero reimplementar estándares tales como ethernet queda más allá del propósito de este máster

Se decide usar mensajes binarios en diferentes canales: siendo que sólo necesitamos dos tipos de mensaje (señal y peligro), con dos mensajes de entrada y dos de salida basta. El problema de las colisiones no existe, está en nuestra mano decidir si queremos que dos o más mensajes simultáneos cuenten como uno, o incluso si queremos requerir dicha condición para tener una activación (modelo neuronal).

Así pues contamos con las siguientes entradas/salidas:

- Una salida LED

- Dos salidas digitales (señal/emergencia)

- Dos entradas digitales (señal/emergencia)

- Una entrada analógica

Esta entrada en un principio es el sensor capacitivo mencionado, más adelante es sustituido por una antena FM en resonancia con el medio para detectar presencia humana en el entorno.

3.3.2. Hardware: elección de plataforma

Se decide implementar los insectos "free form" sobre microcontroladores de tamaño reducido. Sería posible implementar este proyecto de forma 100% analógica, basado en amplificadores operacionales, puertas lógicas y demás componentes básicos, pero la complejidad de la programación física aumentaría con creces, impidiendo iterar de forma sencilla en los parámetros del modelo y sin ninguna ganancia clara.

De entre todos los microcontroladores disponibles se decide empezar a trabajar con STM32F41. Si bien son menos conocidos que la plataforma Arduino, presentan una serie de ventajas que los hacen atractivos a primera vista, a saber:

- Mayor potencia de cálculo
- Marginalmente más económicos
- Sensor capacitivo integrado

Sin embargo, la inconsistencia de los STM32 adquiridos (poder programarlos era un proceso altamente aleatorio, con desconexiones frecuentes y problemas similares) y el hecho de que dos de tres ardiesen sin aviso previo, decantó la balanza por los más conocidos Arduino Nano, a los que se adaptaron los esquemáticos realizados.

3.3.3. Software: simulaciones de sistema

A lo largo del proceso de diseño se han implementado diversas simulaciones para estudiar y determinar diferentes aspectos de la obra final, desde la viabilidad de los modelos propuestos hasta los parámetros idóneos del modelo final seleccionado o la implementación práctica de la obra final.

Las simulaciones de modelos se han programado en Processing por su flexibilidad y herramientas para realizar programación visual, así como la posibilidad de implementar programación orientada a objetos de forma nativa. La programación orientada a objetos[2] es un paradigma de programación en el que la complejidad de un programa se abstrae en componentes modulares que pueden interaccionar entre sí, que tienen un estado concreto y métodos para actuar sobre ese estado, y que pueden reimplementarse tantas veces como sea necesario. Esto es inmensamente útil en un contexto como el de esta pieza: podemos implementar un objeto base (ya sea "torreta", "luciérnaga" o similares) e instanciarlo decenas o cientos de veces para ver como interacciona con el resto.

Processing también plantea problemas serios que ha tocado sortear: si bien permite implementar objetos de manera nativa, no tiene un mecanismo de señales establecido. Las señales son un mecanismo que en el contexto de la programación orientada a objetos permite comunicar unos objetos con otros o responder a ciertos eventos. Siendo que esta es una obra basada en la comunicación entre elementos, ha tocado implementar versiones rústicas de este tipo de mecanismos.

Otro problema que plantea Processing es que todo su paradigma de programación es fuertemente síncrono y ligado a un método que se ejecuta cíclicamente en cada tick de un reloj sobre el que tampoco se tiene demasiado control. Se ha optado por una aproximación pseudo-asíncrona en estas

2 Esta definición no es precisa y deja fuera muchos de los aspectos de la programación orientada a objetos, pero a falta de una base más detallada sobre paradigmas de programación y dado el contexto en que se usa la usaremos libremente.

simulaciones, haciendo que de forma síncrona se modifique el estado de cada objeto pero condicionando la ejecución de ciertos métodos a dicho estado. Con una frecuencia de reloj lo bastante alta se pueden simular eventos asíncronos de forma bastante fidedigna, con la desventaja de que el código es menos legible de lo que sería si se hubiese usado un framework que soportase de forma nativa asincronía, eventos y señales.

Simulaciones del Prototipo 3

La simulación del prototipo final, también por ser la definitiva, es mucho más elaborada. El programa está para ser ejecutado con una interfaz limpia (hasta el punto en que se puede entender como obra artística por sí misma) o con una interfaz de depuración, en la que se muestra información en tiempo real de lo que ocurre con cualquiera de los nodos mostrados. También incluye dos modelos diferentes de interacción, posibilidad de modificar el prototipo en tiempo real, generación aleatoria de prototipos en el arranque, y un modelo basado en la física para simular la organización de los nodos en el grafo.

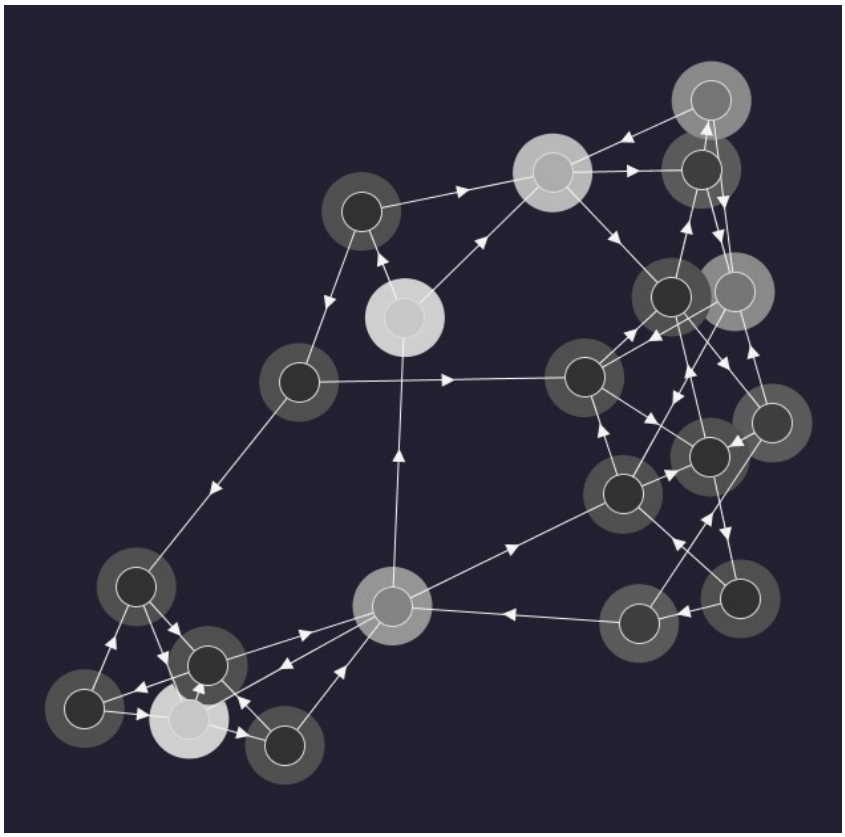


Figura 10: Simulación en curso con nodos en diversos grados de activación

La interfaz consiste principalmente en una serie de nodos representados por dos círculos concéntricos, y conectores que unen dichos nodos en lo que en matemáticas o teoría de la información se conoce como grafo dirigido (es decir, un grafo en el que las conexiones tienen dirección y codifican cómo fluye la información a través de ellas). Los nodos se pueden mover clickando y arrastrando desde el círculo interior, y las conexiones se pueden crear clickando desde el círculo exterior hacia cualquier otro nodo. Mientras se mueve uno de los nodos aparece en la parte inferior de la aplicación una zona demarcada con otro color que permite eliminarlo si se suelta en ella.

Se ha implementado así mismo un simulador basado en física para calcular las posiciones de los nodos a lo largo del tiempo. Una fuerza repele a los nodos entre sí, mientras que los conectores actúan como fuerza atractiva dependiente de la distancia (es decir, similar a un muelle). Ambas fuerzas están calibradas para que el grafo ocupe el máximo posible de la pantalla y otorga una capa de entendimiento a la estructura de este: nodos con muchas conexiones se tienden a desplazar a zonas con mayor densidad de nodos por superficie, mientras que zonas inconexas del grafo se aíslan automáticamente. El motor de física añade además una dimensión de dinamismo que ha sido apreciada por la mayoría de quienes han interactuado con el prototipo.

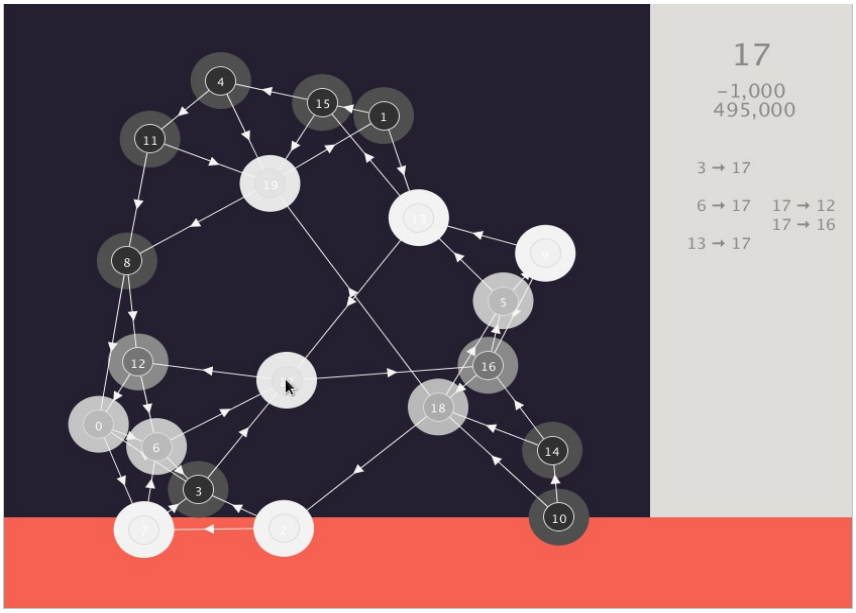


Figura 11: Simulador con la barra de depuración activa y la zona de borrado de nodos activada mientras se arrastra el nodo 17

Al clickar en un nodo este se ilumina con un breve destello, siendo transmitida una señal de alarma a los nodos con los que está conectado por un conector saliente. Por otra parte cada nodo cuenta así mismo con la información de los nodos conectados consigo mismo. Si la interfaz de *debug* está activada disponemos también en el lateral con una zona en la que aparece información del nodo sobre el que está el cursor en cualquier momento.

Cómo reacciona un nodo a los parpadeos o señales de alarma depende del modelo de interacción seleccionado: mientras que el modelo "neuronal" depende de la activación de los nodos colindantes para que se produzca otra activación, en el modelo "influencia" todos los nodos parpadean con una frecuencia propia, que es modificada cuando uno de los nodos a los que está conectado se activa. En ambos modelos hay un factor de fatiga y saturación, y no son capaces de volver a excitarse hasta que pasa un determinado tiempo.

3.3.4. Diseño e implementación

Partiendo de las restricciones descritas comenzamos diseñando los esquemáticos de los módulos e implementándolos en una protoboard, diseñando de forma paralela posibles diseños *freeform* para el apartado escultórico de la pieza. Se realizan modelos 3D de dichos diseños para estudiar su viabilidad artística y se integran en escenas para visualizar cómo se integrarían en una instalación.

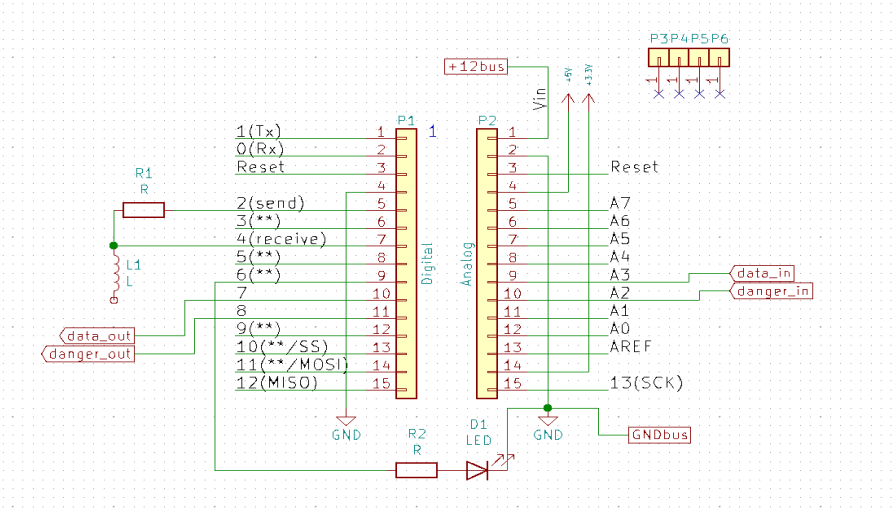


 Figura 12: Esquemáticos iniciales de los dispositivos

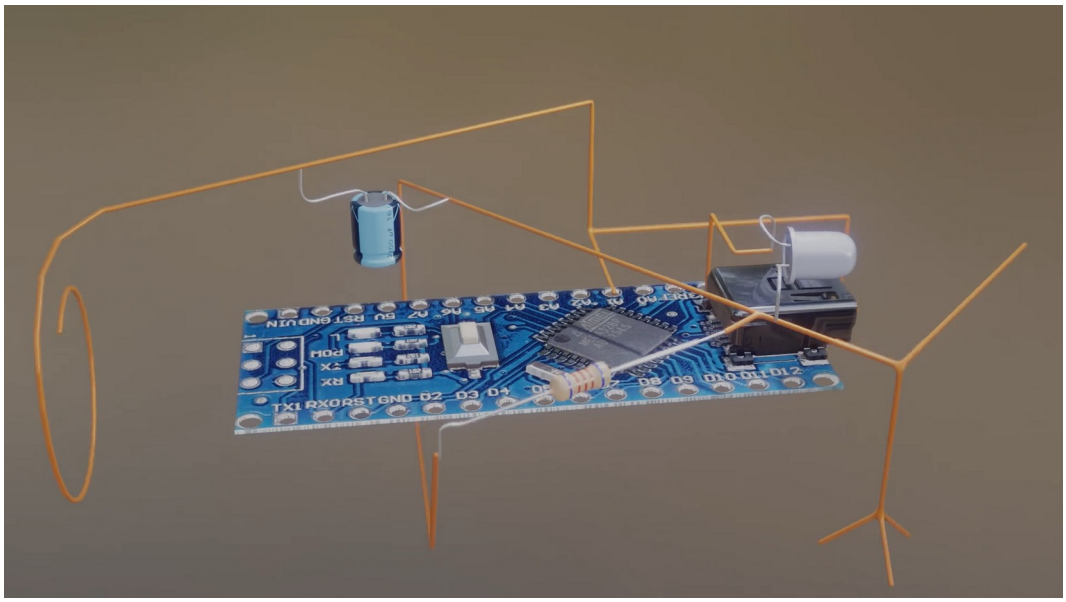


Figura 14: Mockup inicial del dispositivo

Figura 13: Mockup inicial de la instalación

El hecho de cambiar el sensor capacitivo por una antena alargada implica repensar de forma radical el diseño de los módulos. Los componentes que eran necesarios para el sensor capacitivo dejan de ser necesarios, haciendo redundante gran parte de la estructura que había sido planeada. Además, la antena a añadir es especialmente larga porque ha de operar en un rango de frecuencias cuya longitud de onda está en el orden de los 0.5-1 metros (así que ha de ser por lo menos de medio metro de longitud). Por bien que se integre es

un elemento desestabilizador en la estructura, con lo que se deshecha la idea de que los módulos adopten una postura que no sea apoyados completamente en la superficie en la que descansen.

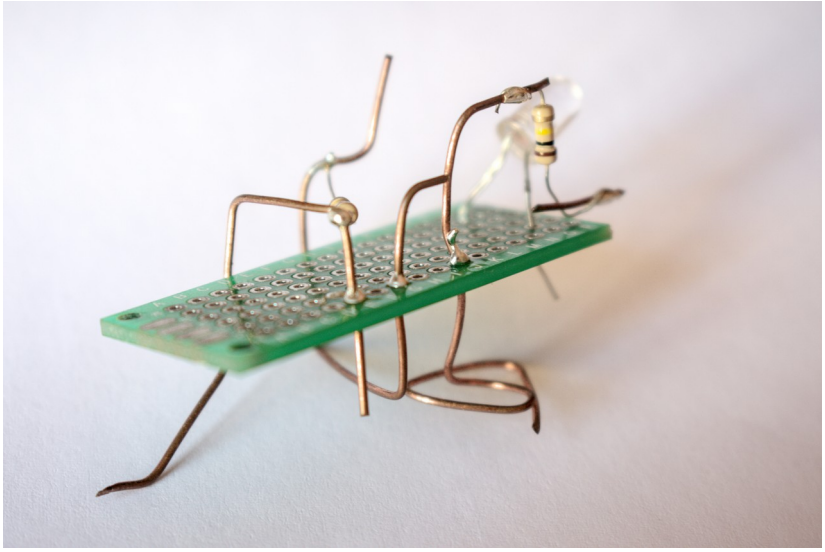


Figura 15: Prototipo inicial con cuerpo free-form capacitivo

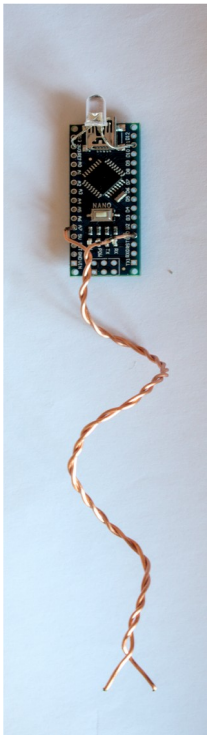


Figura 16: Flagelo en el diseño final

Finalmente se opta porque las antenas -consistentes en un par de cables de cobre barnizados y trenzados- adopten una forma ligeramente helicoidal, recordando el flagelo de múltiples organismos unicelulares. En oposición a dicho flagelo tenemos el emisor LED, que convenientemente dispone en esa posición de un pin de alimentación de 3.3V, con lo que no es necesario añadir resistencias para regular la potencia que lo alimenta. Se opta por usar cable fino aislado para las entradas y salidas, teniendo dos de cada en los lados del dispositivo. Cada salida se conecta a otros 3 dispositivos, siendo necesario un entramado relativamente denso de cables en la obra final, reminiscente de los sistemas neuronales y simbiosis micorriza antes mencionadas.

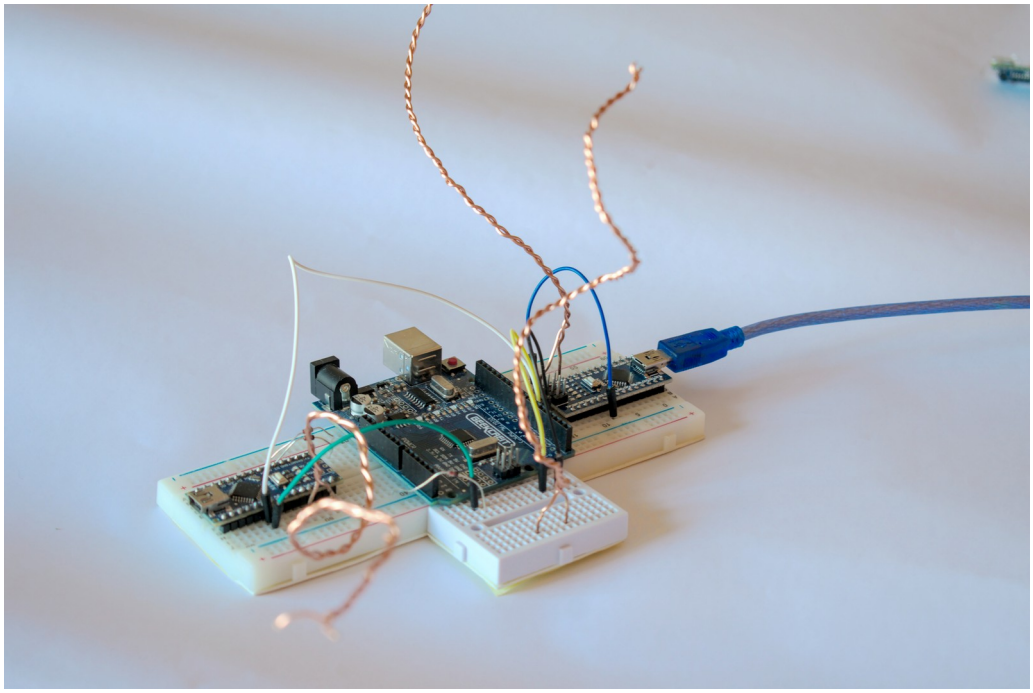


Figura 17: Banco de pruebas para testar interacciones entre las antenas

Se construye un pequeño número de unidades que se programan con código Arduino. Para simplificar el mantenimiento se desarrolla una librería que permite gestionar contadores asíncronos con funciones de *callback* cuando llegan a 0, y modificables externamente, así como una modificación de la librería Buzz (Labs, 2016/2021) que permite utilizar antenas como sensores ambientales de presencia y movimiento. A pesar de las severas limitaciones de Arduino, el uso de estas dos librerías, y el hecho de no tener que gestionar más que el propio dispositivo (y no necesitar referencias del resto), así como la existencia de "señales", en este caso analógicas, para poder comunicarlos entre ellos simplifica en gran medida el código que adaptamos del prototipo implementado anteriormente en Processing.

La obra contempla que se dispongan 20-40 módulos de este tipo interconectados y desplegados en función de las características del espacio expositivo.

3.4 Análisis de los resultados. Discusión

Los resultados de las simulaciones implementadas son altamente satisfactorios, habiendo logrado el tipo de comportamiento emergente. El sistema es percibido como sistema coherente y no como impulsos aleatorios, y es capaz de captar la atención del público tanto si interactúan o no con la instalación. La experiencia ha sido descrita como hipnótica, y se ha hecho énfasis en más de una ocasión en el carácter evolutivo de los patrones que se perciben.

Los primeros prototipos también han mostrado funcionar de acuerdo al diseño planteado, abriendo así la puerta a ejecutar la obra en su totalidad en un futuro. Las estimaciones de coste de realización han subestimado la complejidad de realizar esculturas basadas en cobre, y en general se ha encontrado que producir los módulos es más costoso de lo que inicialmente se esperaba. Siendo una obra que requiere un número elevado de dichos módulos este coste se ha traducido en la imposibilidad de producir la obra en un plazo razonable, lo que ha motivado la no inclusión de la producción de la obra final en los ámbitos del presente trabajo.

4. Conclusiones

Este TFM se enmarca en una investigación más amplia sobre los fenómenos emergentes en el arte y en la sociedad. Se ha podido estudiar este tipo de fenómenos y realizar una investigación extensiva tanto de las implicaciones de modelos alternativos de relación societaria como de los artistas que han abordado esta casuística. Se han detectado patrones en la forma de tratar los fenómenos emergentes en el arte y se ha podido plantear un proyecto que aporta una visión novedosa al ser implementado en su totalidad de forma analógica (considerando como analógico un sistema en el que no se simula por software el comportamiento global de éste). Se abre así la puerta a ejecutar dicha instalación en una fase posterior de la investigación como trabajo futuro, y se asientan las bases teóricas y el discurso general del proyecto.

Respecto a los objetivos específicos planteados, se han analizado las principales características de la noción de emergencia en ecosistemas biológicos y se han establecido procesos similares en el devenir de la instalación propuesta.

También se ha contextualizado el proyecto de instalación artística *Emergencia* en el marco de las prácticas artísticas contemporáneas que comparten nocioes afines.

Por último, se han señalado los puntos fundamentales y necesarios del debate sobre el apoyo mutuo-individualismo, el cual consideramos hoy trascendente ante la deriva de control y sometimiento de la sociedad actual.

De todo ello se deriva la necesidad de analizar los procesos que conlleva la emergencia y colaborar tal como hacen árboles y hongos, entre otros organismos biológicos, para que este sistema social no nos destruya.

5. Fuentes consultadas

Abramovic, V., & Glynn, R. (2018, abril 3). *Edge of Chaos | Interactive Architecture Lab*. http://www.interactivearchitecture.org/lab-projects/edge-of-chaos

Asimov, I. (1966). *Foundation*. Avon.

Bailey, J. (2020). *The Game of Life—Emergence in Generative Art*. Artnome. https://www.artnome.com/news/2020/7/12/the-game-of-life-emergence-in-generative-art

Berenguer, J. M. (2007). *Luci, sin nombre y sin memoria*. https://doi.org/10.13140/2.1.3556.2880

Bernoulli, D. (1738). *Hydrodynamica, sive de viribus et motibus fluidorum commentarii: Opus academicum ab auctore, dum Petropoli ageret, congestum*. http://www.e-rara.ch/zut/1227730

Björkman, E. (1970). Forest tree mycorrhiza—The conditions for its formation and the significance for tree growth and afforestation. *Plant and Soil*, *32*(1), 589-610. https://doi.org/10.1007/BF01372897

Christa Sommerer and Laurent Mignonneau artworks. (s. f.). Recuperado 16 de febrero de 2021, de http://www.interface.ufg.ac.at/christa-laurent/WORKS/FRAMES/FrameSet.html

Colombo, M., & Knauff, M. (2020). Editors' Review and Introduction: Levels of Explanation in Cognitive Science: From Molecules to Culture. *Topics in Cognitive Science*, *12*(4), 1224-1240. https://doi.org/10.1111/tops.12503

Cooper, M. (2016). *Emergence*. https://emergence.maxcooper.net

Copeland, J., & Moiseff, A. (2004). Flash Precision at the Start of Synchrony in Photuris frontalis. *Integrative and Comparative Biology*, *44*(3), 259-263. https://doi.org/10.1093/icb/44.3.259

Darwin, C. (1871). *The Descent of Man, and Selection in Relation to Sex*. J. Murray.

Darwin, C., Josa i LLorca, J., & Zulueta, A. de. (2013). *El origen de las especies*. Espasa Libros.

Francis, R., & Read, D. J. (1984). Direct transfer of carbon between plants connected by vesicular–arbuscular mycorrhizal mycelium. *Nature*, *307*(5946), 53-56. https://doi.org/10.1038/307053a0

Gardner, M. (1970). MATHEMATICAL GAMES. *Scientific American*, *223*(4), 120-123.

Gorzelak, M. A., Asay, A. K., Pickles, B. J., & Simard, S. W. (2015). Inter-plant communication through mycorrhizal networks mediates complex adaptive behaviour in plant communities. *AoB PLANTS*, *7*. https://doi.org/10.1093/aobpla/plv050

Hackaday. (2019, noviembre 17). *Mohit Bhoite—Building Free-Formed Circuit Sculptures*. https://www.youtube.com/watch?v=LqVFxNFGNbc

Helgason, T., & Fitter, A. (2005). The ecology and evolution of the arbuscular mycorrhizal fungi. *Mycologist*, *19*(3), 96-101. https://doi.org/10.1017/S0269-915X(05)00302-2

Hoff, A. (2016). *On Generative Algorithms: Introduction · inconvergent*. Inconvergent.Net. https://inconvergent.net/generative/

Holland, J. H. (1998). *Emergence. From chaos to order*. Perseus Books.

Jones. (s. f.). *Chaos in an Atmosphere Hanging on a Wall | Mathematics of Planet Earth*. Recuperado 11 de septiembre de 2021, de

http://mpe.dimacs.rutgers.edu/2013/03/17/chaos-in-an-atmosphere-hanging-on-a-wall/

Kropotkin, P. A., Orsetti, L., & Montagu, A. (2016). *El apoyo mutuo: Un factor de evolución.* Pepitas de Calabaza.

Labs, L. (2021). *BUZZ library* [C++]. https://github.com/connornishijima/arduino-buzz (Original work published 2016)

Laplace, P. S. (1814). *A Philosophical Essay on Probabilities* (F. L. Emory & F. W. Truscott, Trads.). https://www.gutenberg.org/ebooks/58881

Lozano-Hemmer, R. (2004). *Synaptic Caguamas*. https://lozano-hemmer.com/synaptic_caguamas.php

MARTIN MESSIER | IMPULSE. (s. f.). Recuperado 11 de septiembre de 2021, de https://martinmessier.art/impulse.html

Newman, E. I. (1988). Mycorrhizal Links Between Plants: Their Functioning and Ecological Significance. En M. Begon, A. H. Fitter, E. D. Ford, & A. Macfadyen (Eds.), *Advances in Ecological Research* (Vol. 18, pp. 243-270). Academic Press. https://doi.org/10.1016/S0065-2504(08)60182-8

panGenerator, ◥. (2013, octubre 8). *CONSTELLACTION*. https://vimeo.com/76479685

Reid, C. P., & Woods, F. (1969). *Translocation of C^(14)-Labeled Compounds in Mycorrhizae and It Implications in Interplant Nutrient Cycling*. https://doi.org/10.2307/1934844

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25-34. https://doi.org/10.1145/37402.37406

Sebastian Lague. (2021). *Complex Behaviour from Simple Rules: 3 Simulations*. https://www.youtube.com/watch?v=kzwT3wQWAHE

Simard, S. W., Perry, D. A., Jones, M. D., Myrold, D. D., Durall, D. M., & Molina, R. (1997). Net transfer of carbon between ectomycorrhizal tree species in the field. *Nature*, *388*(6642), 579-582. https://doi.org/10.1038/41557

The /r/place Atlas. (s. f.). Recuperado 11 de septiembre de 2021, de https://draemm.li/various/place-atlas/

Lista de figuras

7. Anexo

7.1 Código de las simulaciones

7.1.1 Prototipo 1

Emergence

```
Obstacle[] walls;
CircleBoundary circle;
Obstacle[] turrets;
Obstacle[] obstacles;
Obstacle person;
//Turret[] turrets;
Ray ray;
float xoff = 0;
float yoff = 10000;

void setup() {
  size(600, 600);


  obstacles = new Obstacle[40+1+4];
  //walls = new Obstacle[turrets.length+4];
  int espacio = 100;
  for (int i = 0; i < obstacles.length; i++) {
    obstacles[i] = new Turret(random(width), random(height));
    //obstacles[i] = new Turret(i*espacio % width, (i*espacio / width)*espacio);
    //turrets[i] = new Turret(random(width), random(height));
    //walls[i] = turrets[i].light_sensor;
  }


  person = new CircleBoundary(0, 0, 20);
  //turrets = (Turret[]) aturrets;

  obstacles[obstacles.length-5] = person;
  obstacles[obstacles.length-4] = (new Boundary(0, 0, width, 0));
  obstacles[obstacles.length-3] = (new Boundary(width, 0, width, height));
  obstacles[obstacles.length-2] = (new Boundary(width, height, 0, height));
  obstacles[obstacles.length-1] = (new Boundary(0, height, 0, 0));

  turrets = (Obstacle[]) subset(obstacles, 0,  obstacles.length-5);
}

void draw() {
  background(255);

  ((CircleBoundary) person).pos = new PVector(mouseX, mouseY);
  person.show();

  for (Obstacle _turret : turrets){
    Turret turret = (Turret) _turret;
    turret.update(obstacles);
    turret.show();
  }
  //particle.update(noise(xoff)*width, noise(yoff)*height);
  //particle.update(mouseX, mouseY);
  //particle.show();
  //particle.look(walls);

  xoff += 0.01;
  yoff += 0.01;
```

```
}
```

Boundary

```
class Boundary implements Obstacle{
  PVector a, b;
  Boundary(float x1, float y1, float x2, float  y2) {
    this.a = new PVector(x1, y1);
    this.b = new PVector(x2, y2);
  }

  void show() {
    //stroke(50);
    line(this.a.x, this.a.y, this.b.x, this.b.y);
    stroke(50);
  }

  void touch() {
    stroke(255, 0, 0);
    this.show();
  }
}
```

CircleBoundary

```
class CircleBoundary implements Obstacle {
  PVector pos;
  float radius;
  CircleBoundary(float x, float y, float _radius) {
    this.pos = new PVector(x, y);
    this.radius = _radius;

  }

  void show() {

    noFill();
    circle(this.pos.x, this.pos.y, this.radius*2);
    stroke(50);
  }

  void touch() {
    stroke(255, 0, 0);
    this.show();
  }
}
```

Obstacle

```
interface Obstacle {
  void show();
  void touch();
}
```

Particle

```
class Particle {
  PVector pos;
  PVector angle;
  float ray_velocity;
  Ray ray;
  Particle(float _x, float _y) {
    this.pos = new PVector(_x, _y);
    this.ray_velocity = random(-0.008,0.008);
    this.angle = PVector.fromAngle(random(2*PI));
    this.ray = new Ray(this.pos, this.angle);

  }

  void update() {
    this.update(this.pos.x, this.pos.y);
  }

  void update(float x, float y) {
    this.pos.set(x, y);
    this.angle.rotate(this.ray_velocity);
  }

  void look(Obstacle[] obstacles) {
    PVector closest = null;
    float record = 500000000;
    Obstacle touched_obstacle = null;
    for (Obstacle obstacle : obstacles) {
      PVector pt = this.ray.cast(obstacle);
      if (pt != null) {
        float d = PVector.dist(this.pos, pt);
        if (d < record) {
          record = d;
          closest = pt;
          touched_obstacle = obstacle;
        }
      }
    }
      if (closest != null) {
        // colorMode(HSB);
        // stroke((i + frameCount * 2) % 360, 255, 255, 50);
        stroke(255, 0, 0, 100);
        line(this.pos.x, this.pos.y, closest.x, closest.y);
        touched_obstacle.touch();
      }

  }

  void show() {
    fill(255, 255,0);
    ellipse(this.pos.x, this.pos.y, 4, 4);
    //this.ray.show();
  }
}
```

Ray

```
class Ray {
  PVector pos, dir;
  Ray(PVector _pos, PVector _angle) {
    this.pos = _pos;
    this.dir = _angle;
  }

  void lookAt(float x, float y) {
    this.dir.x = x - this.pos.x;
    this.dir.y = y - this.pos.y;
    this.dir.normalize();
  }

  void show() {
    stroke(255, 0, 0);
    push();
    translate(this.pos.x, this.pos.y);
    line(0, 0, this.dir.x * 10, this.dir.y * 10);
    pop();
  }



  PVector cast(Obstacle obstacle) {

    if (obstacle instanceof Boundary) {

      Boundary wall = (Boundary) obstacle;

      float x1 = wall.a.x;
      float y1 = wall.a.y;
      float x2 = wall.b.x;
      float y2 = wall.b.y;

      float x3 = this.pos.x;
      float y3 = this.pos.y;
      float x4 = this.pos.x + this.dir.x;
      float y4 = this.pos.y + this.dir.y;

      float den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
      if (den == 0) {
        return null;
      }

      float t = ((x1 - x3) * (y3 - y4) - (y1 - y3) * (x3 - x4)) / den;
      float u = -((x1 - x2) * (y1 - y3) - (y1 - y2) * (x1 - x3)) / den;
      if (t > 0 && t < 1 && u > 0) {
        PVector pt = new PVector();
        pt.x = x1 + t * (x2 - x1);
        pt.y = y1 + t * (y2 - y1);
        return pt;
      } else {
        return null;
      }
    } else if (obstacle instanceof Turret || obstacle instanceof CircleBoundary) {

      // let AB be the ray, with origin in A
      // let  C be the center of the circle
      // let  D be the closest point in AB to C
      CircleBoundary turret = null;
      if (obstacle instanceof CircleBoundary){
        turret = (CircleBoundary) obstacle;
      }else{
        turret = ((Turret) obstacle).light_sensor;
      }
```

```
      PVector AB = this.dir;
      PVector AC = PVector.sub(turret.pos,this.pos);
      PVector AD = PVector.mult(AB.normalize(null),
constrain(PVector.dot(AC,AB.normalize(null)), 0, 100000));

      //stroke(255,0,0);
      //line(this.pos.x, this.pos.y, PVector.add(this.pos,
PVector.mult(AB,1000)).x, PVector.add(this.pos,  PVector.mult(AB,1000)).y);
      //stroke(0,0,255);
      //line(this.pos.x, this.pos.y, PVector.add(this.pos, AC).x,
PVector.add(this.pos, AC).y);
      //stroke(0,255,0);
      //line(this.pos.x, this.pos.y, PVector.add(AD, this.pos).x, PVector.add(AD,
this.pos).y);

      //PVector CD = PVector.sub(AD, PVector.add(this.pos, turret.pos));
      PVector CD = PVector.sub(AD, AC);

      //stroke(0,255,255);
      //line(turret.pos.x, turret.pos.y, PVector.add(CD, turret.pos).x,
PVector.add(CD, turret.pos).y);

      if (AC.mag() < turret.radius) {          // we're inside the circle and we
consider it an oriented surface
        return null;
      } else if (CD.mag() > turret.radius) {   // we're not touching the circle
        return null;
      } else if (CD.mag() == turret.radius) {  // wll, we're touching at one point
        return CD.sub(turret.pos);
      } else {
        float dD = sqrt(sq(turret.radius) - CD.magSq());
        AD.setMag(AD.mag() - dD);

        //stroke(255,0,255);
        //line(this.pos.x, this.pos.y, PVector.add(AD, this.pos).x,
PVector.add(AD, this.pos).y);

        return PVector.add(AD,this.pos);
      }
    } else {

      return null;
    }
  }
}
```

Turret

```
class Turret implements Obstacle{
  PVector pos;
  CircleBoundary light_sensor;
  Particle laser;
  float laser_velocity;
  Ray ray;
  Turret(float _x, float _y) {
    this.pos = new PVector(_x, _y);
    this.light_sensor = new CircleBoundary(this.pos.x, this.pos.y, 10);
    this.laser = new Particle(this.pos.x, this.pos.y);
    this.laser_velocity = random(-0.000,0.008);

  }

  void update(Obstacle[] obstacles) {
    //this.pos.set(x, y);
```

```
    this.laser.update();
    this.laser.look(obstacles);

    this.laser.ray_velocity = this.laser_velocity;
  }


  void show() {
    fill(255, 255,0);
    this.laser.show();
    this.light_sensor.show();
    //this.ray.show();
  }

  void touch(){
    stroke(0, 0, 255);
    this.laser.ray_velocity = 0.03;
    //this.show();

  }
}
```

7.1.2 Prototipo 2

Emergence2

```
int numBalls = 60;
int selectedBall = -1;
Ball[] balls = new Ball[numBalls];

void setup(){
  size(640, 360);
  fill(255, 204);
  stroke(100);

  for (int i=0; i<numBalls; i++){
    balls[i] = new Ball(random(width), random(height), 20, i, balls);
  }
}

void draw(){
  background(0);
  for (Ball ball : balls) {
    ball.check();
    ball.move(selectedBall);
    ball.display();
  }
}

void mousePressed(){
  for (int i=0; i<numBalls; i++){
    if (balls[i].selected(mouseX, mouseY)){
      selectedBall = i;
      break;
    }
    if (balls[i].stimulated(mouseX, mouseY)){
      balls[i].lit = true;
    }
  }
}

void mouseReleased(){
  selectedBall = -1;
}
```

Ball

```
boolean debug = true;

class Ball{
  float x, y;
  float diameter;
  int id;
  boolean lit;
  float sensibility;
  float cooldown = 10;
  float remaining_cooldown;
  float max_brightness;
  float min_brightness;
  float brightness;
  Ball[] others;

  Ball(float _x, float _y, float _diam, int _id, Ball[] _others) {
    x = _x;
    y = _y;
    diameter = _diam;
    id = _id;
    lit = false;
    sensibility = 50;
    remaining_cooldown = 0;
    max_brightness = 30;
    min_brightness = 0;
    brightness = 0;
    others = _others;
  }

  void check(){
    if (this.lit){
      this.lit = false;
      this.brightness = max_brightness;
      this.remaining_cooldown = cooldown;
    }

    if (this.brightness > min_brightness){
      this.brightness -= 2;
    }

    if (this.remaining_cooldown>0){
      this.remaining_cooldown -= 1;
    }else{
      for (Ball other : others) {
        if (this.id != other.id){
          this.lit = this.lit || ((dist(this.x, this.y, other.x, other.y) <
(this.sensibility + other.brightness)) && (other.brightness != 0));
        }
      }
    }
  }

  void move(int ball) {
    if (this.id == ball){
      this.x = mouseX;
      this.y = mouseY;
    }

  }

  boolean selected(float _x, float _y) {
    float distance = dist(_x, _y, x, y);
    return (distance < diameter);
  }
```

```
  boolean stimulated(float _x, float _y) {
    float distance = dist(_x, _y, this.x, this.y);
    return ((diameter < distance) && (distance < sensibility));
  }

  void display() {
    float _intensity = map(this.brightness, min_brightness, max_brightness, 0,
255);
    fill(_intensity);
    ellipse(x, y, diameter, diameter);
    if(debug){
      noFill();
      stroke(255);
      ellipse(x, y, sensibility*2, sensibility*2);
      stroke(0,255,0);
      ellipse(x, y, brightness*2, brightness*2);
    }
  }
}
```

7.1.3 Prototipo 3

Emergence3

```
boolean random_graph = true;
boolean debug = false;

int window_width = 640;
int window_height = 640;
int debug_width = 200;

int numBalls = 0;
int selectedBall = -1; //-1 if nothing, else the id number
int extSelectedBall = -1;
ArrayList<Ball> balls = new ArrayList<Ball>();
Ball ball;
ArrayList<Connector> connectors = new ArrayList<Connector>();

int lastBallId = 0;

class SortedBall implements Comparable<SortedBall> {
  Ball ball;
  float distance;

  SortedBall(Ball ball1, Ball ball2) {
    this.ball = ball2;
    this.distance = dist(ball1.x, ball1.y, ball2.x, ball2.y);
  }

  public int compareTo(SortedBall c0) {
    return (int) Math.signum(distance - c0.distance);
  }
}

void settings() {


  if (debug) {
    size(window_width+debug_width, window_height);
  } else {
    size(window_width, window_height);
  }

  smooth(8);
}
```

```
void setup() {
  frameRate(60);


  noStroke();

  if (random_graph) {
    drawRandomGraph();
  }
}

float custom_sort (Ball ball1, Ball ball2) {

  return (dist(ball1.x, ball1.y, ball2.x, ball2.y));
}

void draw() {
  background(#241f31);

  fill(#deddda);
  rect(window_width, 0, width, height);
  relaxGraph();


  if (selectedBall != -1) {
    fill(#f66151);
    noStroke();
    rect(0, height*0.85, width, height);
  }

  for (Ball ball : balls) {
    ball.check();
    if (selectedBall != -1) {
      ball.move(selectedBall);
    }

    if (ball.selected(mouseX, mouseY)) {
      int last_y = 50;
      textAlign(CENTER, CENTER);
      textSize(32);
      text(ball.id, 740, last_y);

      textAlign(CENTER, CENTER);
      textSize(20);
      if (ball.in_danger){
        last_y += 40;
        text("in danger", 740, last_y);
      }

      last_y += 40;
      text(ball.remaining_countdown, 740, last_y);

      last_y += 20;
      text(ball.remaining_cooldown, 740, last_y);

      last_y += 50;

      int i = 0;
      for (Ball parent : ball.parents) {
        textAlign(RIGHT, BASELINE);
        textSize(16);
        last_y += 20;
        text(parent.id + " → " + ball.id, 740, last_y+i*20);
        i++;
      }

      i=0;

      for (Ball child : ball.children) {
```

```
        textAlign(LEFT, BASELINE);
        textSize(16);
        text(ball.id + " → " + child.id, 760, last_y+i*20);
        i++;
      }
    }
    ball.outer_display();
  }

  for (Ball ball : balls) {
    ball.update_status();
  }

  for (Connector connector : connectors) {
    connector.update();
    connector.display();
  }

  for (Ball ball : balls) {
    ball.inner_display();
  }

  if (extSelectedBall != -1) {
    ball = balls.get(extSelectedBall);
  }


  //noLoop();
}

void mousePressed() {
  for (Ball ball : balls) {
    if (ball.selected(mouseX, mouseY)) {
      selectedBall = balls.indexOf(ball);
      ball.set_in_danger();
      break;
    }

    if (ball.ext_selected(mouseX, mouseY)) {
      extSelectedBall = balls.indexOf(ball);
      connectors.add( new Connector(ball));
      break;
    }
  }
  if (selectedBall == -1 && extSelectedBall == -1) {
    ball = new Ball(mouseX, mouseY, 30, balls.size(), balls, connectors);
    balls.add(ball);
  }
}

void mouseReleased() {
  for (Ball ball : balls) {
    if (ball.selected(mouseX, mouseY)) {
      selectedBall = balls.indexOf(ball);
      break;
    }

    if (ball.ext_selected(mouseX, mouseY)) {
      selectedBall = balls.indexOf(ball);
      break;
    }
  }
  if (extSelectedBall != -1) {
    connectors.remove(connectors.size()-1); //remove last element

    if (selectedBall != -1) {
      connectors.add( new Connector(balls.get(extSelectedBall),
balls.get(selectedBall)));
```

```
      for (Ball ball : balls) {
        ball.update_neighbours();
      }
    }
  } else {
    // remove balls  by dropping them at the bottom
    if (selectedBall != -1) {
      if (mouseY > height*0.85) {


        Ball ballToRemove = balls.get(selectedBall);
        ArrayList<Connector> connectorsToRemove = new ArrayList<Connector>();

        //lets check all the connectors to remove any references to this ball
        for (Connector connector : connectors) {
          if (connector.ball1 == ballToRemove || connector.ball2 == ballToRemove) {
            connectorsToRemove.add(connector);
          }
        }

        connectors.removeAll(connectorsToRemove);
        balls.remove(balls.get(selectedBall));

        for (Ball ball : balls) {
          ball.update_neighbours();
        }
      }
    }
  }

  selectedBall = -1;
  extSelectedBall = -1;
}

void drawRandomGraph() {
  for (int i=0; i<numBalls; i++) {
    balls.add(new Ball(random(window_width*0.70) + window_width*0.15,
      random(window_height*0.70) + window_height*0.15,
      30, i, balls, connectors));
  }

  ArrayList<SortedBall> sorted_balls = new ArrayList<SortedBall>();
  ArrayList<SortedBall> sorted_balls_to_delete = new ArrayList<SortedBall>();

  for (int i=0; i<numBalls; i++) {

    // for each ball we sort the other ones by distance
    // naturally the ball 0 will be itself

    sorted_balls.clear();

    for (int j=0; j<numBalls; j++) {
      sorted_balls.add(new SortedBall(balls.get(i), balls.get(j)));
    }

    java.util.Collections.sort(sorted_balls);

    sorted_balls.get(0).ball.update_neighbours();

    // we want to remove some of them
    sorted_balls_to_delete.clear();

    for (SortedBall sorted_ball : sorted_balls) {
      // like, if they're already have too many parents
      if (sorted_ball.ball.parents.size() >= 3) {
        sorted_balls_to_delete.add(sorted_ball);
      }

      // or if $this is a children of it
```

```
      for (Ball sorted_ball_child: sorted_ball.ball.children){
        if (sorted_balls.get(0).ball.id == sorted_ball_child.id){
          sorted_balls_to_delete.add(sorted_ball);
        }
      }
    }

    sorted_balls.removeAll(sorted_balls_to_delete);
    sorted_balls.remove(sorted_balls.get(0));

    for (int k=0; k<2; k++) {
      if (k<sorted_balls.size())
        connectors.add( new Connector(balls.get(i), sorted_balls.get(k).ball));
    }

    for (Ball ball : balls) {
      ball.update_neighbours();
    }
  }
}

void relaxGraph() {
  //get desired medium distance
  float min_length = Float.MAX_VALUE;
  float max_length = Float.MIN_VALUE;

  for (Connector connector : connectors) {
    float connector_length = dist(connector.x, connector.y, connector.x2,
connector.y2);
    if (connector_length < min_length)
      min_length = connector_length;
    if (connector_length > max_length)
      max_length = connector_length;
  }

  //float desired_length = (min_length + max_length) / 2;
  float desired_length = 30;

  ArrayList<Ball> balls_snapshot = new ArrayList<Ball>(balls);
  for (int i = 0; i < balls.size(); i++) {
    PVector attracting_vector = new PVector(0, 0);

    if (balls.get(i).is_linked) {
      ArrayList<Ball> attracting_balls = new
ArrayList<Ball>(balls_snapshot.get(i).parents);
      attracting_balls.addAll(balls_snapshot.get(i).children);
      for (Ball attracting_ball : attracting_balls) {
        PVector v1 = new PVector(attracting_ball.x-balls_snapshot.get(i).x,
attracting_ball.y-balls_snapshot.get(i).y);
        float current_length = v1.mag();
        attracting_vector.add(v1.normalize().mult((current_length-
desired_length)*0.1));
      }

      PVector repelling_vector = new PVector(0, 0);
      ArrayList<Ball> repelling_balls = new ArrayList<Ball>(balls_snapshot);
      repelling_balls.remove(balls_snapshot.get(i));
      for (Ball unlinked_ball : balls_snapshot) {
        if (unlinked_ball.is_linked == false)
          repelling_balls.remove(unlinked_ball);
      }

      for (Ball repelling_ball : repelling_balls) {
        PVector v1 = new PVector(repelling_ball.x-balls_snapshot.get(i).x,
repelling_ball.y-balls_snapshot.get(i).y);
        float current_length = v1.mag();
        repelling_vector.sub(v1.normalize().mult(300/pow(current_length, 1)));
      }
```

```
      attracting_vector.add(repelling_vector);
    }

    if (balls.get(i).x < 100)
      attracting_vector.add(new PVector(0-balls.get(i).x + 100, 0).mult(0.1));
    if (balls.get(i).x > window_width - 100)
      attracting_vector.add(new PVector(window_width - balls.get(i).x- 100,
0).mult(0.05));
    if (balls.get(i).y < 100)
      attracting_vector.add(new PVector(0, 0-balls.get(i).y + 100).mult(0.1));
    if (balls.get(i).y > (window_height - 100))
      attracting_vector.add(new PVector(0, window_height - balls.get(i).y -
100).mult(0.05));

    balls.get(i).x += attracting_vector.x;
    balls.get(i).y += attracting_vector.y;
  }
}
```

Ball

```
boolean influenza = false;
boolean neuron = true;

class Ball {
  float x, y;
  float diameter;
  float selection_diameter;
  int id;

  boolean lit;
  boolean last_lit;

  boolean in_danger;
  boolean last_in_danger;

  float cooldown = 10;
  float remaining_cooldown;
  float remaining_countdown;
  int danger_cooldown;
  boolean countdown_active = false;
  float max_brightness;
  float min_brightness;
  float brightness;
  float period;  //in seconds
  ArrayList<Ball> others;
  ArrayList<Connector> connectors;
  ArrayList<Ball> parents;
  ArrayList<Ball> children;
  boolean is_linked;



  Ball(float _x, float _y, float _diam, int _id, ArrayList<Ball> _others,
ArrayList<Connector> _connectors) {
    x = _x;
    y = _y;
    diameter = _diam;
    selection_diameter = diameter * 2;
    id = _id;
    lit = 0.2 > random(1);
    last_lit = lit;
    remaining_cooldown = 0;
    remaining_countdown = -1;
    max_brightness = 30;
    min_brightness = 0;
    brightness = 0;
```

```
    period = random(2, 10);
    others = _others;
    connectors = _connectors;

    parents = new ArrayList<Ball>();
    children = new ArrayList<Ball>();

    //this.update_neighbours();
  }

  void check() {
    if (this.lit) {
      this.lit = false;
      this.brightness = max_brightness;

      if (influenza && !this.in_danger) {
        this.remaining_cooldown = period*60;
      }

      if (neuron && !this.in_danger) {
        this.remaining_cooldown = 0;
      }

      this.last_in_danger = false;
      if (this.in_danger) {
        this.last_in_danger = true;
        this.in_danger = false;
      }
    }

    if (this.brightness > min_brightness) {
      this.brightness -= 2;
    }

    if (this.remaining_countdown < 0 && this.countdown_active == true) {
      this.lit = true;
      this.countdown_active = false;
    }

    if (this.remaining_countdown >= 0) {
      this.remaining_countdown -= 1;
    }

    if (this.danger_cooldown > 0) {
      this.danger_cooldown -= 1;
    }


    if (this.remaining_cooldown>0) {
      this.remaining_cooldown -= 1;
      if (influenza) {
        for (Ball parent : parents) {
          if (parent.lit == true) {
            this.remaining_cooldown = this.remaining_cooldown/2;
          }
        }
      }
    } else {
      if (influenza)
        this.lit = true;
      if (neuron) {
        for (Ball parent : parents) {
          if (parent.last_lit == true) {
            this.remaining_cooldown = 0;
            this.remaining_countdown = period*10;
            this.countdown_active = true;
            //this.lit = true;
          }
        }
```

```
    }
  }
  for (Ball parent : parents) {
    if (parent.last_in_danger == true && this.danger_cooldown == 0) {
      this.set_in_danger();
    }
  }
}

void set_in_danger() {
  this.remaining_cooldown = 500;
  this.danger_cooldown = 100;
  this.remaining_countdown = 1;
  this.countdown_active = true;
  this.in_danger = true;
}

void move(int ball) {
  if (others.get(ball) == this) {
    this.x = mouseX;
    this.y = mouseY;
  }
}

void update_status() {
  this.last_lit = this.lit;
  this.id = this.others.indexOf(this);
}

void update_neighbours() {

  this.parents.clear();
  this.children.clear();
  for (Connector connector : connectors) {
    if (connector.ball2 == this) {
      this.parents.add(connector.ball1);
    }
    if (connector.ball1 == this) {
      this.children.add(connector.ball2);
    }
  }

  if (this.parents.size()+this.children.size() == 0) {
    this.is_linked = false;
  } else {
    this.is_linked = true;
  }
}

boolean selected(float _x, float _y) {
  float distance = dist(_x, _y, x, y);
  return (distance < diameter/2);
}

boolean ext_selected(float _x, float _y) {
  float distance = dist(_x, _y, x, y);
  return (diameter/2 < distance && distance < selection_diameter/2);
}


void inner_display() {
  float _intensity = map(this.brightness, min_brightness, max_brightness, 50, 255);
  fill(_intensity);
  stroke(#deddda);
  ellipse(x, y, diameter, diameter);
  if (debug) {
    fill(255);
    text(this.id, this.x, this.y);
  }
```

```
  }

  void outer_display() {
    float _intensity = map(this.brightness, min_brightness, max_brightness, 80,
255);
    fill(_intensity);
    textAlign(CENTER, CENTER);
    textSize(12);
    noStroke();
    ellipse(x, y, selection_diameter, selection_diameter);
    //filter(BLUR, 1);
  }
}
```

Connector

```
class Connector{
  Ball ball1, ball2;
  float x, y, x2, y2;
  boolean temporary = true;
  int id;

  Connector(Ball ball1) {
    x = ball1.x;
    y = ball1.y;
    x2 = mouseX;
    y2 = mouseY;
    this.ball1 = ball1;
    this.ball2 = null;
    temporary = true;
  }

  Connector(Ball ball1, Ball ball2) {
    x = ball1.x;
    y = ball1.y;
    x2 = ball2.x;
    y2 = ball2.y;
    this.ball1 = ball1;
    this.ball2 = ball2;
    temporary = false;
  }

  void update() {
    this.x = ball1.x;
    this.y = ball1.y;

    if (temporary){
      this.x2 = mouseX;
      this.y2 = mouseY;
    }else{
      this.x2 = ball2.x;
      this.y2 = ball2.y;
    }
  }

  void display() {
    stroke(#f6f5f4);
    line(this.x, this.y, this.x2, this.y2);

    PVector middle = new PVector((x2-x)/2 + x,(y2-y)/2 + y);
    PVector normal = new PVector(y2-y,x-x2).normalize(null);
    PVector forward = new PVector((x2-x)/2,(y2-y)/2).normalize(null);

    PVector p1 = PVector.add(middle, PVector.mult(forward,10));
    PVector p2 = PVector.add(middle, PVector.mult(normal,5));
    PVector p3 = PVector.sub(middle, PVector.mult(normal,5));
```

```
    noStroke();
    fill(#f6f5f4);

    triangle(p1.x, p1.y, p2.x, p2.y, p3.x, p3.y);
  }
}
```

7.2 Código de Arduino

7.2.1 Librerías

Código de la librería implementada para adaptar el código de la Simulación del Prototipo 3 a arduino. Permite implementar cronómetros concurrentes

Timer.c

```
#include "Timer.h"
Timer::Timer() {

}
void Timer::begin(int duration, callbackFunc func) {
  this->duration = duration;
  this->callback = func;
  this->running = false;
}
void Timer::start() {
  if (running == false) {
    this->startTime = millis();
    this->endTime = this->startTime + this->duration;
    running = true;
  }
}
void Timer::tick() {
  if (running){
    if (millis() >= this->endTime){
      running = false;
      if (this->callback != NULL)
        this->callback();
    }
  }
}

int Timer::getRemainingTime(){
  return (this->endTime - millis());
}

void Timer::setRemainingTime(int remainingTime){
  this->endTime = millis() + remainingTime;
}
```

Timer.h

```
#ifndef TIMER_H
#define TIMER_H
#include <Arduino.h>
class Timer {

  private:
    int duration;
    long startTime;
    long endTime;
    long remainingTime;

  public:
```

```
    using callbackFunc = void (*)();
    Timer();
    callbackFunc callback;
    bool running = false;
    void begin(int duration, callbackFunc func);
    void start();
    void tick();
    void setRemainingTime(int remainingTime);
    int getRemainingTime();
};
#endif
```