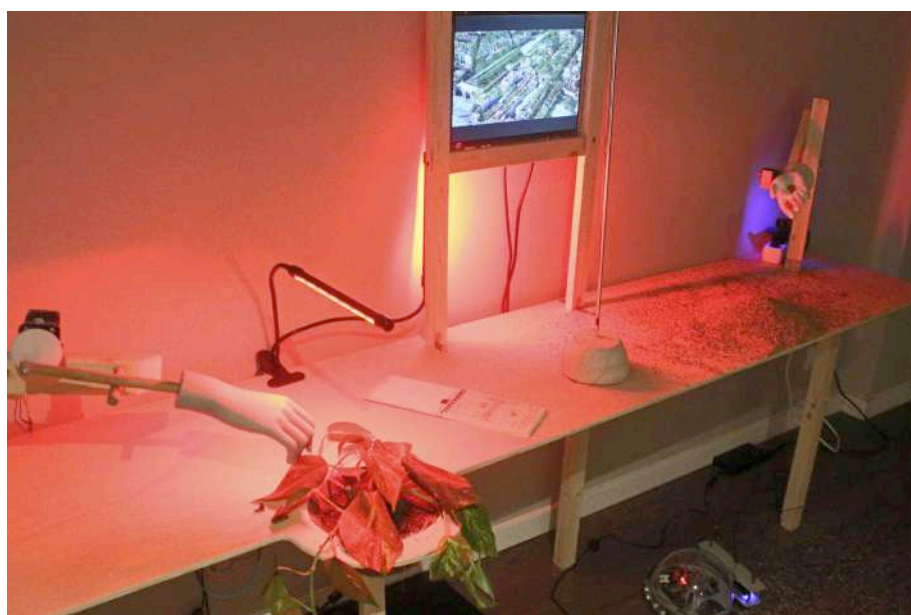


**El entorno que habito. Prototipados artísticos especulativos como
ensayo para la coexistencia interespecie tecnovegetal**

ANEXOS



Trabajo de fin de Máster
Máster Universitario en Artes Visuales y Multimedia

Natacha Cabellos Ricart.
Dirigido por: Dr. Moisés Mañas Carbonell.



Proyecto financiado por Fondart, convocatoria 2019

Valencia, Febrero de 2022.

Índice de anexos

Código de programación a destacar / Prototipado II: Hay que llenarlo todo de plantas (2020)	3
Código Arduino para obtener valores de sensor CO2 y enviarlos por el puerto remoto 10000 asociado a <i>Touchdesigner</i>	3
Código de programación a destacar / Prototipado III: Hacia la conquista de territorios hostiles (2021)	5
Código para obtener los valores de un sensor infrarrojo y publicar el mensaje en el Broker del servidor MQTT.....	5
Código para recibir mensaje desde el Broker del servidor MQTT y activar un servo motor.....	6
Bitácora de trabajo	8

Código de programación a destacar / Prototipado II: Hay que llenarlo todo de plantas (2020)

Código Arduino para obtener valores de sensor CO2 y enviarlos por el puerto remoto 10000 asociado a *Touchdesigner*.

```
1  /*
2   Código para obtener los valores de ppm de un sensor de CO2 MHZ19 y enviar el valor
3   por protocolo OSC a Touchdesigner
4   Este código está adaptado del proyecto grupal "Karensansui: Mudtrack meditation"
5   desarrollado por Atila López de Aguilera, Eduardo Vega y Natacha Cabellos.
6   https://karensansui.wixsite.com/mudcrackmeditation
7   */
8
9 //Incluimos las librerías necesarias
10 #include <Arduino.h>
11 #include <SoftwareSerial.h>
12 #include <ESP8266WiFi.h>
13 #include <WiFiUdp.h>
14 #include <OSCMessages.h>
15 #include <OSCBundles.h>
16 #include <ArduinoOSCETHER.h>
17 #include <ArduinoOSCWifi.h>
18
19 //Define puertos RX y TX en ESP8266
20 #define MH_Z19_RX 12
21 #define MH_Z19_TX 13
22
23 int co2ppm; // CO2 valor medido en ppm
24 SoftwareSerial co2Serial(13, 12); //Asocia el sensor a los pines 13 y 12 del ESP8266
25
26
27 WiFiUDP Udp; // Instancia UDP para mandar y recibir paquetes de datos UDP
28 const IPAddress outIp(192,168,1,152); // IP remota de ordenador
29 const unsigned int outPort = 10000; // Puerto remoto OSC (puerto que recibe)
30 const unsigned int localPort = 8888; // Puerto local OSC
31
32 //Comunicación con el sensor de CO2 MH-Z19
33 int readCO2()
34 {
35 //solicita valores de CO2 y temperatura
36 byte cmd[9] = {0xFF, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79};
37 char antwort[9];
38 co2Serial.write(cmd, 9);
39 // El flujo serial puede desincronizarse.
40 //La respuesta comienza con 0xff, intentar resincronizar.
41 while (co2Serial.available() > 0 && (unsigned char)co2Serial.peek() != 0xFF)
42 {
43 co2Serial.read();
44 }
45 memset(antwort, 0, 9);
46 int i = 0;
47 while (co2Serial.available() == 0)
48 if (co2Serial.available() > 0)
49 {
50 co2Serial.readBytes(antwort, 9);
51 }
```

```

52   byte crc = 0;
53   for (int i = 1; i < 8; i++)
54   {
55     crc += antwort[i];
56   }
57   crc = 255 - crc + 1;
58   if (antwort[0] != char(0xFF))
59   {
60     return -1;
61   }
62   if (antwort[1] != char(0x86))
63   {
64     return -1;
65   }
66   if (antwort[8] == crc)
67   {
68     int antwortHigh = (int) antwort[2];           // respuesta CO2 Byte alto
69     int antwortLow = (int) antwort[3];           // respuesta CO2 Byte bajo
70     int ppm = (256 * antwortHigh) + antwortLow;
71     return ppm;                                   // Respuesta de la ecuación anterior en ppm
72   }
73   else
74   {
75     return -1;
76   }
77   byte status = antwort[5];
78   if (status != 0x40)
79   {
80     Serial.println("Status NOT ok");
81   }
82 }
83
84 char ssid[] = "MIWIFI_7Cfn";           // WIFI SSID
85 char pass[] = "TrfHUGNF";             // Clave WIFI
86
87 //Inicio de la aplicación
88 void setup() {
89   co2Serial.begin(9600);                //Puerto lectura de sensor MH-Z19
90   Serial.begin(115200);
91
92   // Conectar a WIFI
93   WiFi.begin(ssid, pass);
94   while (WiFi.status() != WL_CONNECTED) {
95     delay(500);
96   }
97   Serial.println("WiFi connected");
98   Serial.println("IP address: 192.168.1.152");
99   Serial.println(WiFi.localIP());
100  Serial.println("Starting UDP");
101  Udp.begin(localPort);
102  Serial.print("Local port: ");
103 #ifndef ESP32
104   Serial.println(localPort);
105 #else
106   Serial.println(Udp.localPort());
107 #endif
108 }
109
110 //Enviamos el valor del sensor por protocolo OSC al puerto antes definido
111 void loop() {
112   int co2ppm = readCO2();
113   float vSensor = co2ppm;
114
115   OSCMessage msg("/CO2");
116   msg.add(vSensor);
117   Udp.beginPacket(outIp, outPort);
118   msg.send(Udp);
119   Udp.endPacket();
120   msg.empty();
121   delay(500);
122 }

```

Código de programación a destacar / Prototipado III: Hacia la conquista de territorios hostiles (2021)

Código Arduino para obtener los valores de un sensor infrarrojo y publicar el mensaje en el Broker del servidor MQTT.

```
1  /*
2   Código para obtener los valores de un sensor infrarrojo ("1" o "0") y publicar el mensaje en el
3   Broker del servidor MQTT.
4   Este código fue realizado con la ayuda de Nicolás Mardones.
5   */
6
7  //Librerias
8  #include <ESP8266WiFi.h>
9  #include <PubSubClient.h>
10
11  WiFiClient espClient;          //Instancia de la libreria WiFi llamada espClient
12  PubSubClient client(espClient); //Instancia de la libreria MQTT que usa la instancia de Wifi llamada espClient
13
14  //Variables de conexión
15  const char* ssid = "MIWIFI_7Cfn";          // WIFI SSID
16  const char* password = "TrfhUGNF";       // Clave WIFI
17  const char* mqtt_server = "test.mosquitto.org"; // Servidor MQTT
18
19  //Variables generales
20  unsigned long lastMsg = 0;
21  const int IR_PIN = 14; //SENSOR INFRARROJO
22
23  // Función para conexión a red WIFI
24  void setup_wifi() {
25    delay(10);
26    WiFi.mode(WIFI_STA);          //Configura WiFi como estación.
27    WiFi.begin(ssid, password);    //inicia la conexión con las variables de nombre y clave
28    while (WiFi.status() != WL_CONNECTED) { //Mientras no esté conectado, genera un tiempo para volver a conectar
29      delay(500);
30    }
31    randomSeed(micros());
32  }
33
34  // Función callback es la que recibe el mensaje MQTT y lo procesa.
35  void callback(char* topic, byte* payload, unsigned int length) {
36  }
37
38  //Función de Conexión a servidor MQTT
39  void reconnect() {
40    //Bucle hasta que se reconecta
41    while (!client.connected()) {
42      if (client.connect("espnatacha102")) { //Numero de ID debe ser único en la red
43        client.publish("NatachaCabellos", "Conectado"); //Publica en la sala el mensaje una única vez
44        client.subscribe("NatachaCabellos"); //Suscripción a la sala "NatachaCabellos"
45      } else {
46        delay(100); //retraso para reintentar la conexión
47      }
48    }
49  }
50
51  void setup() {
52    pinMode(IR_PIN, INPUT);
53    setup_wifi();
54    client.setServer(mqtt_server, 1883); //inicia conexión mqtt con puerto por defecto
55    client.setCallback(callback); //configura la función callback para el cliente mqtt
56  }
57
```

```

58 void loop() {
59
60   if (!client.connected()) {           //Si se cae la conexión, se reconecta
61     reconnect();
62   }
63   client.loop();
64
65   int IR = digitalRead(IR_PIN);
66   float IRVal = IR ;
67
68   // Contador
69   unsigned long now = millis();
70   if (now - lastMsg > 500) {
71     lastMsg = now;
72     //Si el valor del sensor es igual o mayor que 1 publica 1, sino, publica 0.
73     if (IRVal >=1){
74       client.publish("NatachaCabellos", "1");
75     } else {
76       client.publish("NatachaCabellos", "0");
77     }
78   }
79 }

```

Código Arduino para recibir mensaje desde el Broker del servidor MQTT y activar un servo motor.

```

1  /*
2   Código para recibir mensaje desde el Broker del servidor MQTT y activar un servo motor.
3   Este código fue realizado con la ayuda de Nicolás Mardones.
4   */
5
6   //Librerías
7   #include <ESP8266WiFi.h>
8   #include <PubSubClient.h>
9   #include <Servo.h>
10
11  //Instancia de la librería Servo
12  Servo myservo;           //Instancia de la librería Servo
13
14  WiFiClient espClient;   //Instancia de la librería WiFi llamada espClient
15  PubSubClient client(espClient); //Instancia de la librería MQTT que usa la instancia de Wifi llamada espClient
16
17  //Variables de conexión
18  const char* ssid = "MIWIFI_7Cfn";           // SSID WIFI
19  const char* password = "TrfHUGNF";        // Clave WIFI
20  const char* mqtt_server = "test.mosquitto.org"; // Servidor MQTT
21
22  //Variables generales
23  int pos = 0;           // variable posición del servo
24  const int pinServo = 12;
25
26  // Función para conexión a red WIFI
27  void setup_wifi() {
28    delay(10);
29    WiFi.mode(WIFI_STA);           //Configura WiFi como estación.
30    WiFi.begin(ssid, password);    //inicia la conexión con las variables de nombre y clave
31    while (WiFi.status() != WL_CONNECTED) { //Mientras no esté conectado, genera un tiempo para volver a conectar
32      delay(500);
33    }
34    randomSeed(micros());

```

```

35 }
36
37 // Función callback es la que recibe el mensaje MQTT y lo procesa.
38 void callback(char* topic, byte* payload, unsigned int length) {
39 // si recibe 1, el servo gira de 0 a 180°, tiene delay de 15 milisegundo
40 if ((char)payload[0] == '1') {
41     for (pos = 0; pos <= 180; pos += 1) {
42         myservo.write(pos);
43         delay(15);
44     }
45     //si recibe otro valor diferente de 1, vuelve a su posición
46 } else {
47     pos = 0;
48     myservo.write(pos);
49 }
50 }
51
52 //Función de Conexión a servidor MQTT
53 void reconnect() {
54 //Bucle hasta que se reconecta
55 while (!client.connected()) {
56     if (client.connect("esnatacha101")) { //Numero de ID debe ser único en la red
57         client.publish("NatachaCabellos", "Conectado"); //Publica en la sala el mensaje una única vez
58         client.subscribe("NatachaCabellos"); //Suscripción a la sala "NatachaCabellos"
59     } else {
60         delay(100); //retraso para reintentar la conexión
61     }
62 }
63 }
64
65 void setup() {
66     setup_wifi();
67     client.setServer(mqtt_server, 1883); //inicia conexión mqtt con puerto por defecto
68     client.setCallback(callback); //configura la función callback para el cliente mqtt
69     myservo.attach(pinServo); //adjunta la variable pinServo a la instancia Servo
70 }
71
72 void loop() {
73     if (!client.connected()) { //Si se cae la conexión, se reconecta
74         reconnect();
75     }
76     client.loop(); //mantiene la conexión abierta
77 }


```

Bitácora de trabajo

Bocetos de estudio para desarrollar dibujos generativos de plantas

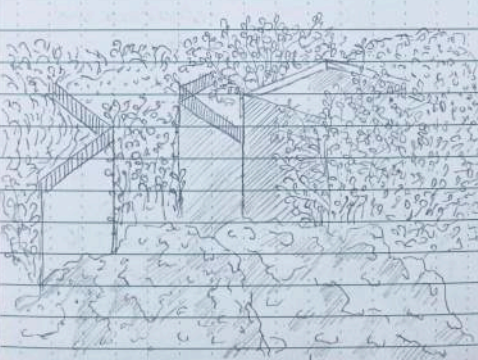
LA FOTOGRAMETRÍA ME PERMITE TIRAR "SCANEAR" O TRASPASAR UN OBJETO NATURAL A 3D DE LA REALIDAD A DIGITAL. LAS PRUEBAS CON LA DIEDRA FUERON BUENAS, SE LOGRÓ ALCANZAR EL VOLUMEN Y LA TEXTURA, PERO NO FUE POSIBLE DIGITALIZAR LA PARTE DE ABAJO / PARA LAS OTRAS PRUEBAS SE DEBE PONER LA PIEDRA SOBRE UN TUBO "BOMBILLA" QUE LA SUJETE SÓLO DESDE EL CENTRO Y PERMITA VER LA PIEDRA EN SU TOTALIDAD PARA REALIZAR EL DIBUJO 360°.

DEFINIR TIPO DE PLANTAS QUE SERÁN ESCANEADAS Y ESCANEAR TRASPASAR A 3D AL IGUAL QUE SELECCIONAR DEL GOOGLE MAPS LAS



LINDADES QUE SE UTILIZARÁN PARA EL FONDO DEL DIBUJO 2D GENERATIVO.

QUIZÁS DEBERÍAN DE SER ARBORES MÁS QUE PLANTAS, PORQUE TIENEN QUE SER COHERENTE CON EL PAISAJE - CIUDAD ESCOGIDA POR CONTEXTO DE DONDE SERÁ EXPUESTO EL PROYECTO.



Rita in the Rain

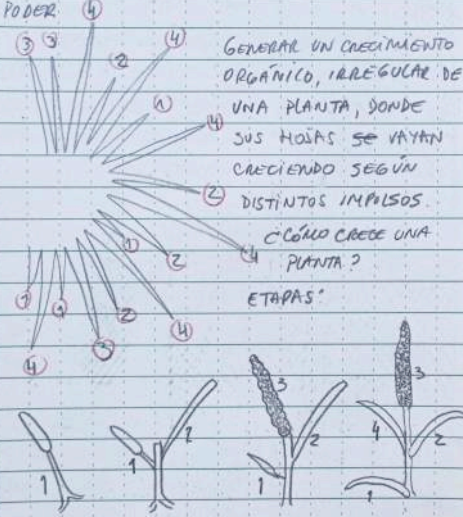
CHOQUE DE GEOMETRÍAS

* RECHAZO DE LA GEOINGENIERÍA COMO IMPOSICIÓN DE PODER.

GENERAR UN CRECIMIENTO ORGÁNICO, IRREGULAR DE UNA PLANTA, DONDE SUS HOJAS SE VAYAN CRECIENDO SEGÚN DISTINTOS IMPULSOS

¿CÓMO CRECE UNA PLANTA?

ETAPAS:



ARBORES ENEMIGOS VIBRICA

PINO

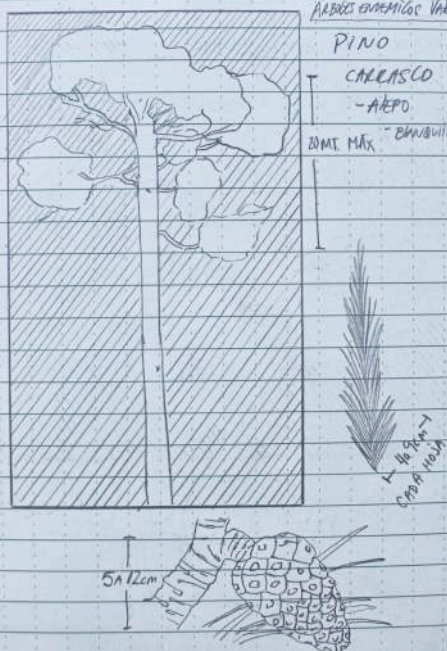
CALLASCO

- AEPD

20MT MÁX - ENBUILO

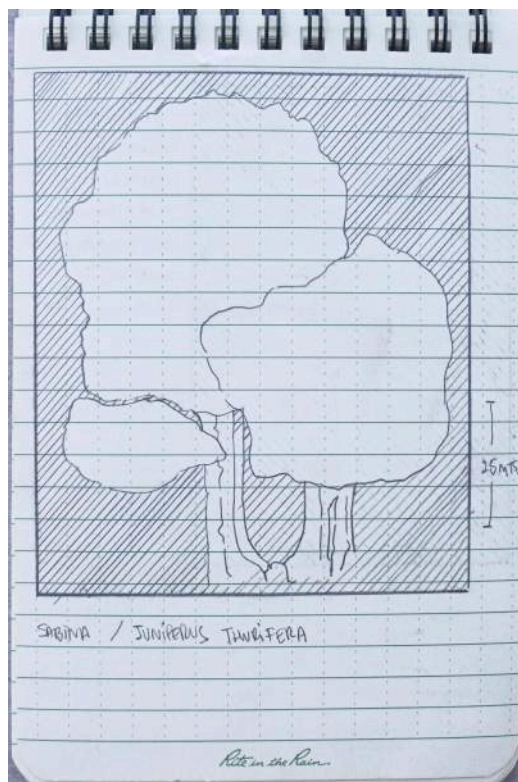
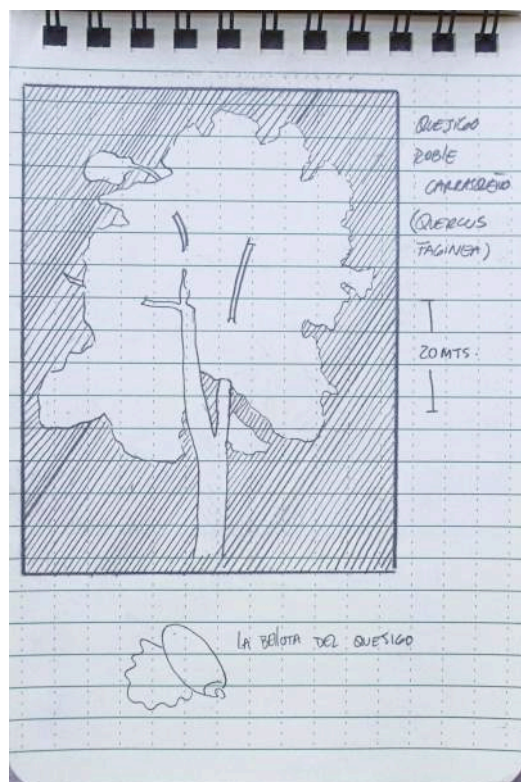
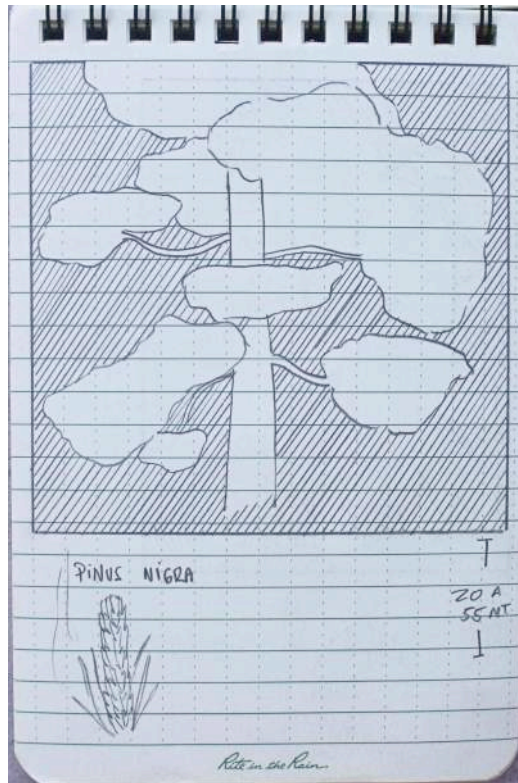
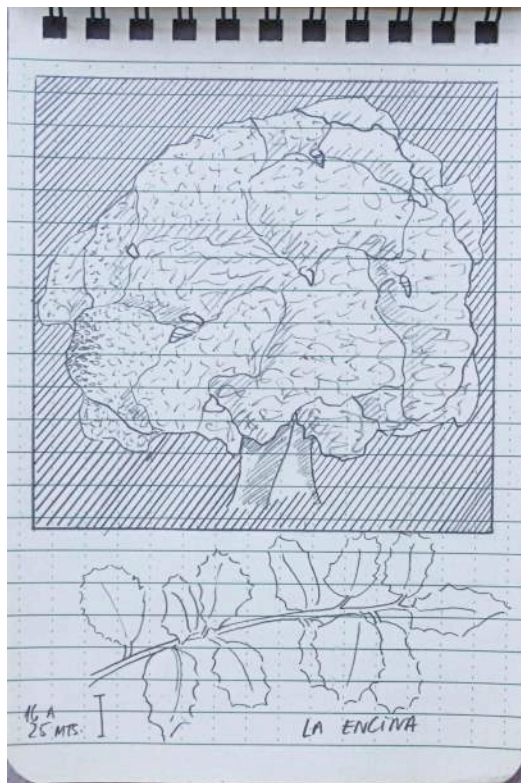
14.9cm - CAPA HOJA

5.12cm

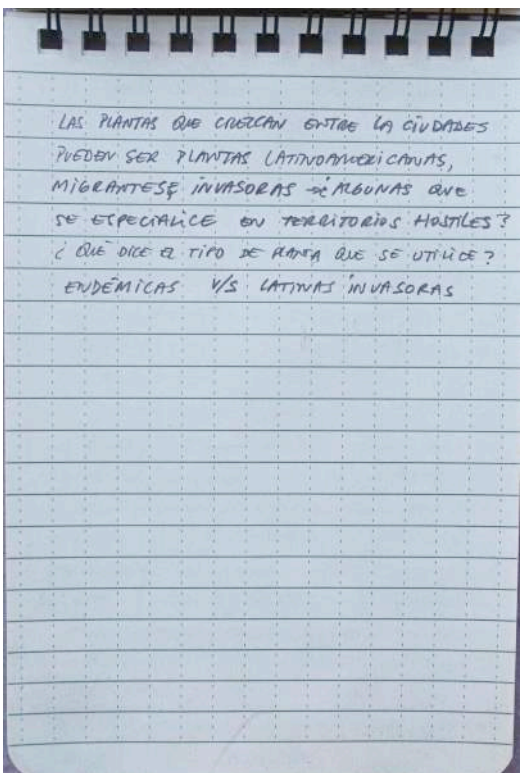
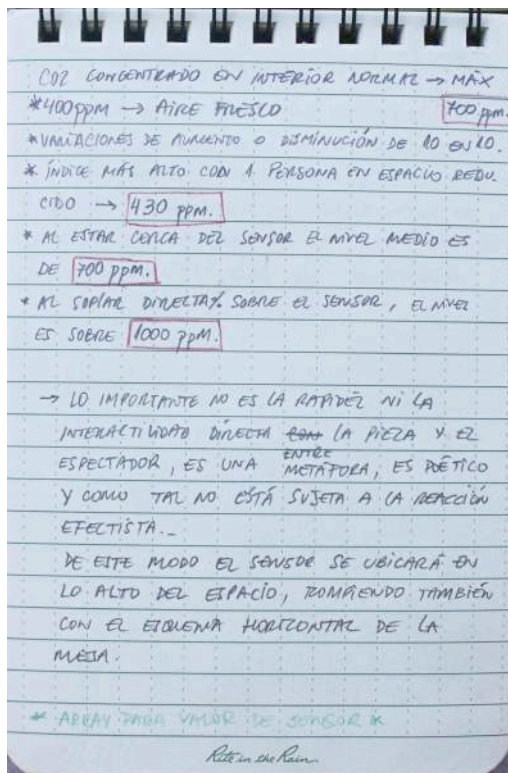
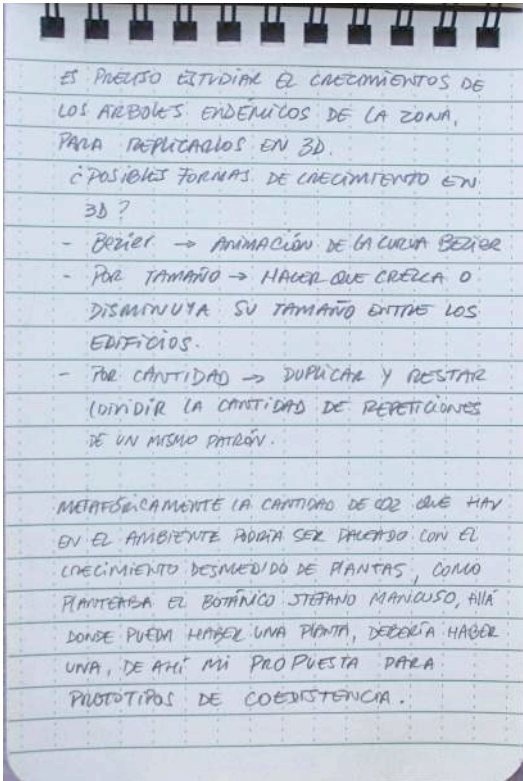
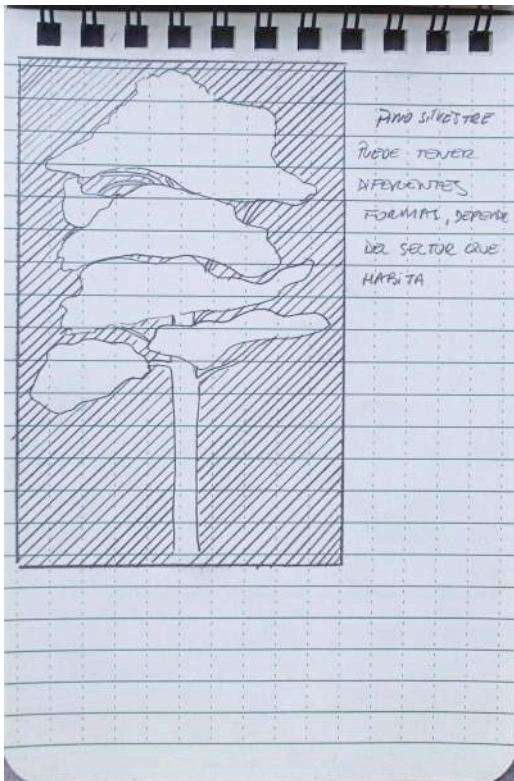


Rita in the Rain

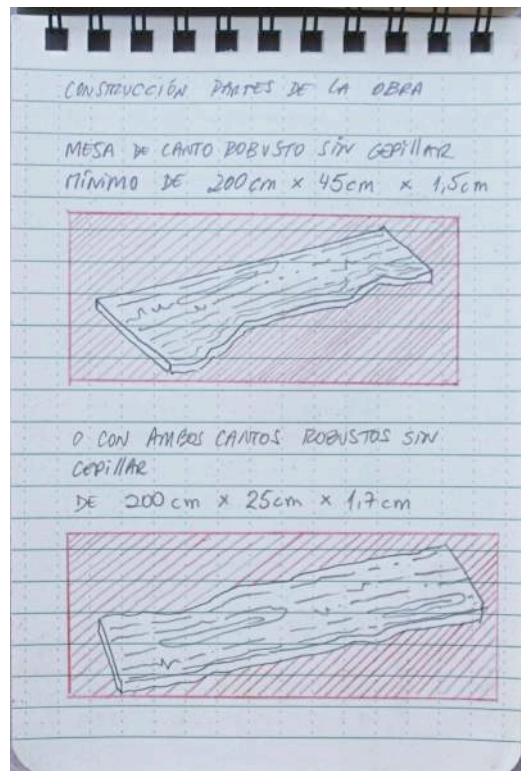
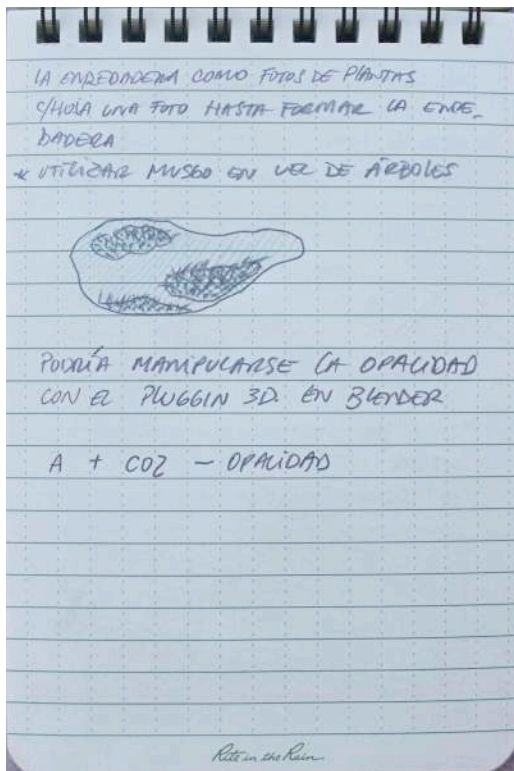
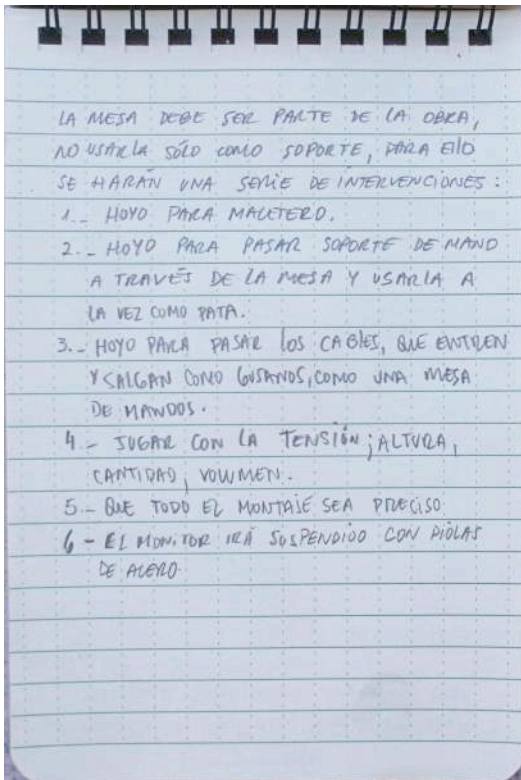
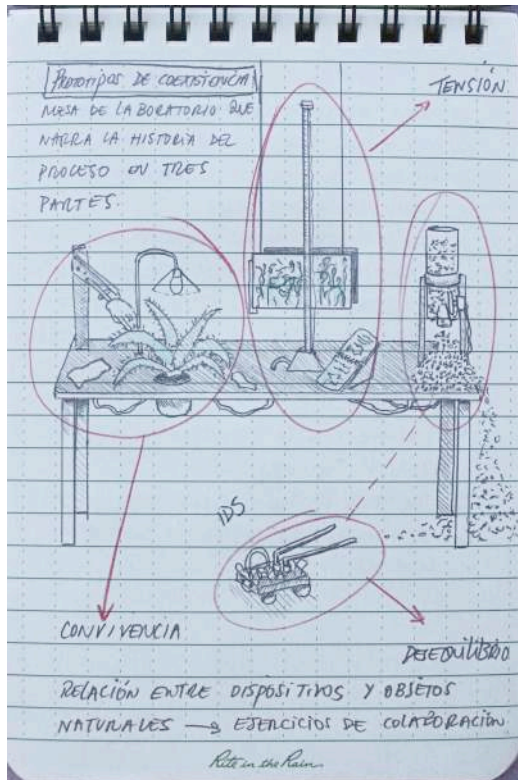
Bocetos de anatomía de plantas endémicas de la Comunidad Valenciana



Anotaciones para la planificación del prototipo II



Propuestas para la construcción de la serie



Bocetos y esquemas técnicos de los prototipados

