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)

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

```
/*
   Código para obtener los valores de ppm de un sensor de CO2 MHZ19 y enviar el valor
   por protocolo OSC a Touchdesigner
   Este código está adaptado del proyecto grupal "Karensansui: Mudtrack meditation"
   desarrollado por Atila López de Aguileta, Eduardo Vega y Natacha Cabellos.
   https://karensansui.wixsite.com/mudcrackmeditation
 */

//Incluimos las librerias necesarias
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <OSCMessage.h>
#include <OSCBundle.h>
#include <ArduinoOSCEther.h>
#include <ArduinoOSCWiFi.h>

//Define puertos RX y TX en ESP8266
#define MH_Z19_RX 12
#define MH_Z19_TX 13

int co2ppm;                           // CO2 valor medido en ppm
SoftwareSerial co2Serial(13, 12);     //Asocia el sensor a los pines 13 y 12 del ESP8266


WiFiUDP Udp;                              // Instancia UDP para mandar y recibir paquetes de datos UDP
const IPAddress outIp(192,168,1,152);     // IP remota de ordenador
const unsigned int outPort = 10000;       // Puerto remoto OSC (puerto que recive)
const unsigned int localPort = 8888;      // Puerto local OSC

//Comunicación con el sensor de CO2 MH-Z19
int readCO2()
{
  //solicita valores de CO2 y temperatura
  byte cmd[9] = {0xFF, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79};
  char antwort[9];
  co2Serial.write(cmd, 9);
  // El flujo serial puede desincronizarse.
  //La respuesta comienza con 0xff, intentar resincronizar.
  while (co2Serial.available() > 0 && (unsigned char)co2Serial.peek() != 0xFF)
  {
    co2Serial.read();
  }
  memset(antwort, 0, 9);
  int i = 0;
  while (co2Serial.available() == 0)
  if (co2Serial.available() > 0)
  {
    co2Serial.readBytes(antwort, 9);
  }
```

```
  byte crc = 0;
  for (int i = 1; i < 8; i++)
  {
    crc += antwort[i];
  }
  crc = 255 - crc + 1;
  if (antwort[0] != char(0xFF))
  {
    return -1;
  }
  if (antwort[1] != char(0x86))
  {
    return -1;
  }
  if (antwort[8] == crc)
  {
    int antwortHigh = (int) antwort[2];            // respuesta CO2 Byte alto
    int antwortLow = (int) antwort[3];             // respuesta CO2 Byte bajo
    int ppm = (256 * antwortHigh) + antwortLow;
    return ppm;                                    // Respuesta de la ecuación anterior en ppm
  }
  else
  {
    return -1;
  }
  byte status = antwort[5];
  if (status != 0x40)
  {
  Serial.println("Status NOT ok");
  }
}

char ssid[] = "MIWIFI_7Cfn";       // WIFI SSID
char pass[] = "TrfHUGNF";          // Clave WIFI

//Inicio de la aplicación
void setup() {
  co2Serial.begin(9600);           //Puerto lectura de sensor MH-Z19
  Serial.begin(115200);

    // Conectar a WIFI
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: 192.168.1.152");
    Serial.println(WiFi.localIP());
    Serial.println("Starting UDP");
    Udp.begin(localPort);
    Serial.print("Local port: ");
#ifdef ESP32
    Serial.println(localPort);
#else
    Serial.println(Udp.localPort());
#endif
}

//Enviamos el valor del sensor por protocolo OSC al puerto antes definido
void loop() {
    int co2ppm = readCO2();
    float vSensor = co2ppm;

    OSCMessage msg("/CO2");
      msg.add(vSensor);
      Udp.beginPacket(outIp, outPort);
      msg.send(Udp);
      Udp.endPacket();
      msg.empty();
      delay(500);
}
```

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.

```
/*
   Código para obtener los valores de un sensor infrarrojo ("1" o "0") y publicar el mensaje en el
   Broker del servidor MQTT.
   Este código fue realizado con la ayuda de Nicolás Mardones.
 */

//Librerías
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient espClient;            //Instancia de la librería WiFi llamada espClient
PubSubClient client(espClient);  //Instancia de la librería MQTT que usa la instancia de Wifi llamada espClient

//Variables de conexión
const char* ssid = "MIWIFI_7Cfn";               // WIFI SSID
const char* password = "TrfHUGNF";              // Clave WIFI
const char* mqtt_server = "test.mosquitto.org";  // Servidor MQTT

//Variables generales
unsigned long lastMsg = 0;
const int IR_PIN = 14; //SENSOR INFRARROJO

// Función para conexión a red WIFI
void setup_wifi() {
  delay(10);
  WiFi.mode(WIFI_STA);                    //Configura WiFi como estación.
  WiFi.begin(ssid, password);             //inicia la conexión con las varibles de nombre y clave
  while (WiFi.status() != WL_CONNECTED) { //Mientras no esté conectado, genera un tiempo para volver a conectar
    delay(500);
  }
  randomSeed(micros());
}

// Función callback es la que recibe el mensaje MQTT y lo procesa.
void callback(char* topic, byte* payload, unsigned int length) {
}

//Función de Conexión a servidor MQTT
void reconnect() {
  //Bucle hasta que se reconecta
  while (!client.connected()) {
    if (client.connect("espnatacha102")) {            //Numero de ID debe ser único en la red
      client.publish("NatachaCabellos", "Conectado"); //Publica en la sala el mensaje una única vez
      client.subscribe("NatachaCabellos");            //Suscripción a la sala "NatachaCabellos"
    } else {
      delay(100);                                     //retraso para reintentar la conexión
    }
  }
}

void setup() {
  pinMode(IR_PIN, INPUT);
  setup_wifi();
  client.setServer(mqtt_server, 1883);    //inicia conexión mqtt con puerto por defecto
  client.setCallback(callback);           //configura la función callback para el cliente mqtt
}
```

```
void loop() {

  if (!client.connected()) {            //Si se cae la conexión, se reconecta
    reconnect();
  }
  client.loop();

  int IR = digitalRead(IR_PIN);
  float IRVal = IR ;

// Contador
  unsigned long now = millis();
  if (now - lastMsg > 500) {
    lastMsg = now;
    //Si el valor del sensor es igual o mayor que 1 publica 1, sino, publica 0.
    if (IRVal >=1){
      client.publish("NatachaCabellos", "1");
    } else {
      client.publish("NatachaCabellos", "0");
    }
  }
}
```

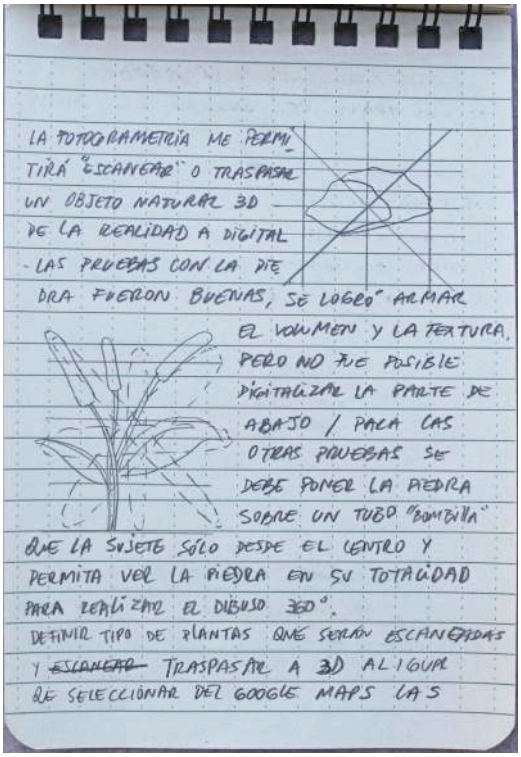
Código Arduino para recibir mensaje desde el Broker del servidor MQTT y activar un servo motor.

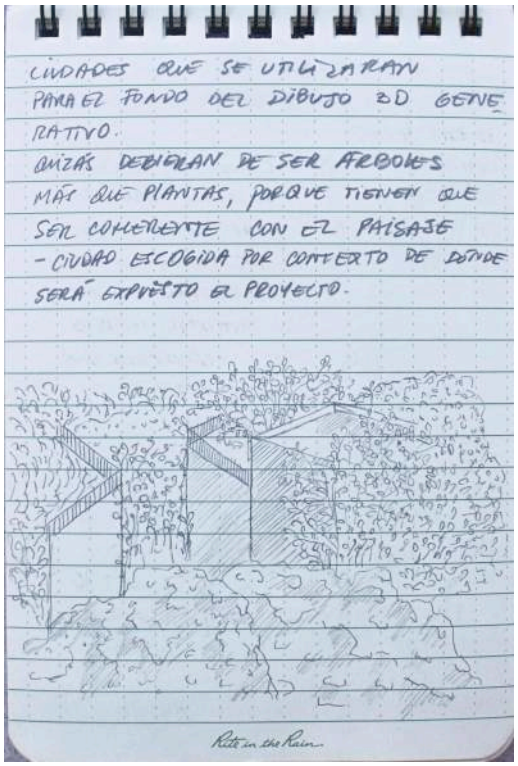
```
/*
   Código para recibir mensaje desde el Broker del servidor MQTT y activar un servo motor.
   Este código fue realizado con la ayuda de Nicolás Mardones.
 */

//Librerias
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Servo.h>

//Instancia de la libreria Servo
Servo myservo;                      //Instancia de la libreria Servo

WiFiClient espClient;               //Instancia de la libreria WiFi llamada espClient
PubSubClient client(espClient);     //Instancia de la libreria MQTT que usa la instancia de Wifi llamada espClient

//Variables de conexión
const char* ssid      = "MIWIFI_7Cfn";            // SSID WIFI
const char* password  = "TrfHUGNF";               // Clave WIFI
const char* mqtt_server = "test.mosquitto.org";   // Servidor MQTT

//Variables generales
int pos = 0;                    // variable posición del servo
const int pinServo = 12;

// Función para conexión a red WIFI
void setup_wifi() {
  delay(10);
  WiFi.mode(WIFI_STA);                  //Configura WiFi como estación.
  WiFi.begin(ssid, password);           //inicia la conexión con las varibles de nombre y clave
  while (WiFi.status() != WL_CONNECTED) { //Mientras no esté conectado, genera un tiempo para volver a conectar
    delay(500);
  }
  randomSeed(micros());
```

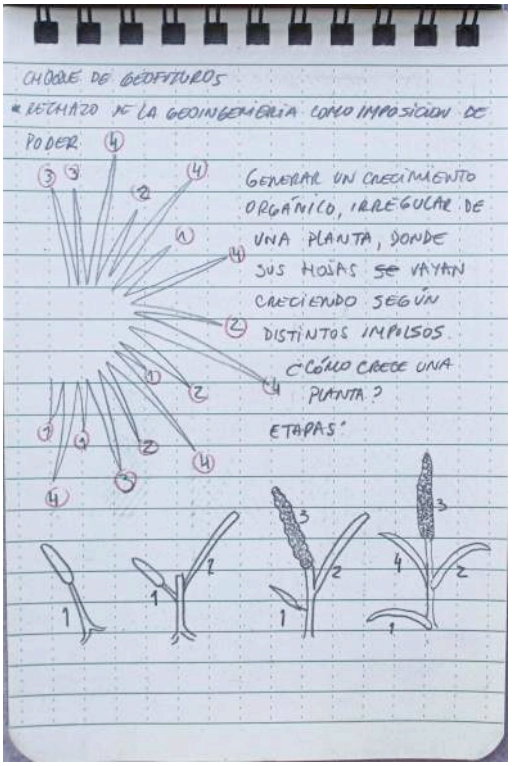
```
}

// Función callback es la que recibe el mensaje MQTT y lo procesa.
void callback(char* topic, byte* payload, unsigned int length) {
  // si recibe 1, el servo gira de 0 a 180°, tiene delay de 15 milisegundo
  if ((char)payload[0] == '1') {
    for (pos = 0; pos <= 180; pos += 1) {
      myservo.write(pos);
      delay(15);
    }
    //si recibe otro valor diferente de 1, vuelve a su posición
  } else {
    pos = 0;
    myservo.write(pos);
  }
}

//Función de Conexión a servidor MQTT
void reconnect() {
  //Bucle hasta que se reconecta
  while (!client.connected()) {
    if (client.connect("espnatacha101")) {            //Numero de ID debe ser único en la red
      client.publish("NatachaCabellos", "Conectado"); //Publica en la sala el mensaje una única vez
      client.subscribe("NatachaCabellos");            //Suscripción a la sala "NatachaCabellos"
    } else {
      delay(100); //retraso para reintentar la conexión
    }
  }
}

void setup() {
  setup_wifi();
  client.setServer(mqtt_server, 1883);    //inicia conexión mqtt con puerto por defecto
  client.setCallback(callback);           //configura la función callback para el cliente mqtt
  myservo.attach(pinServo);               //adjunta la variable pinServo a la instancia Servo
}

void loop() {
  if (!client.connected()) {              //Si se cae la conexión, se reconecta
    reconnect();
  }
  client.loop();                          //mantiene la conexión abierta
}
```

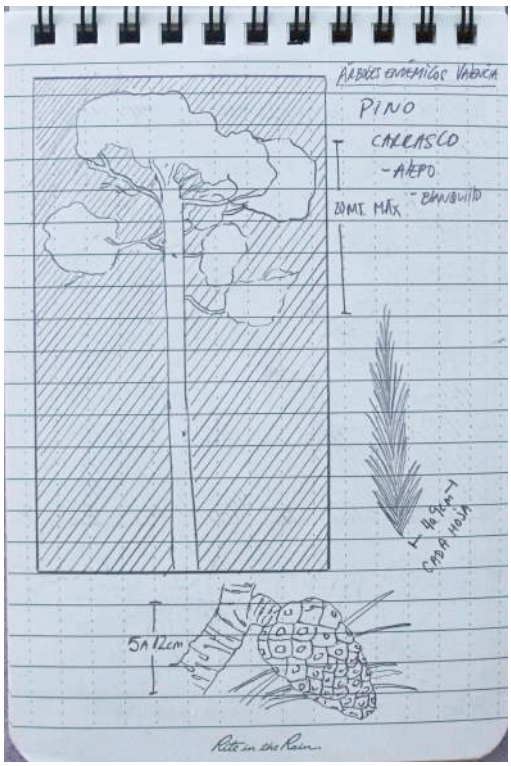
Bitácora de trabajo

Bocetos de estudio para desarrollar dibujos generativos de plantas



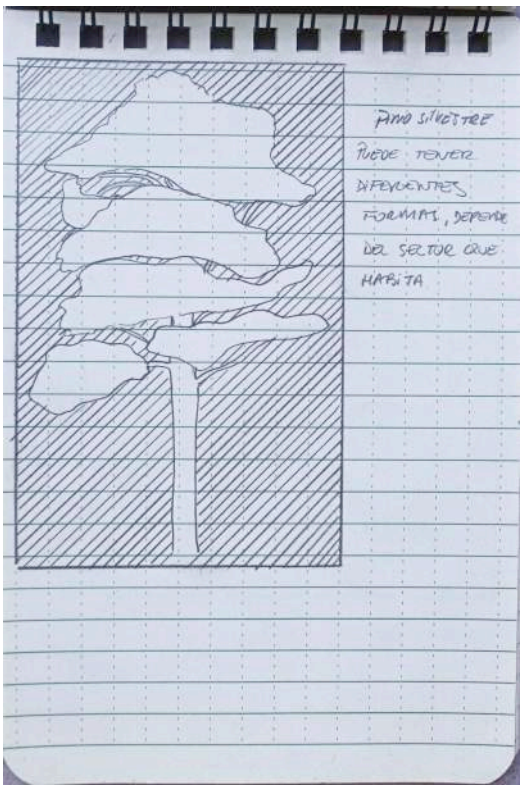


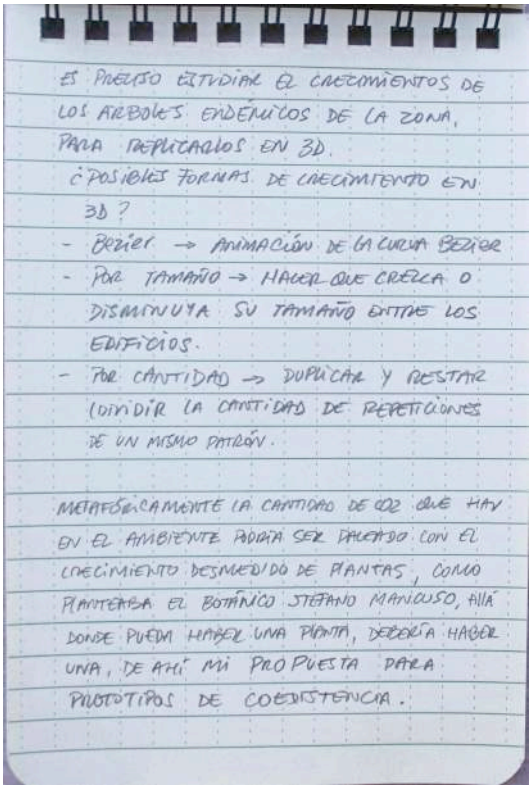


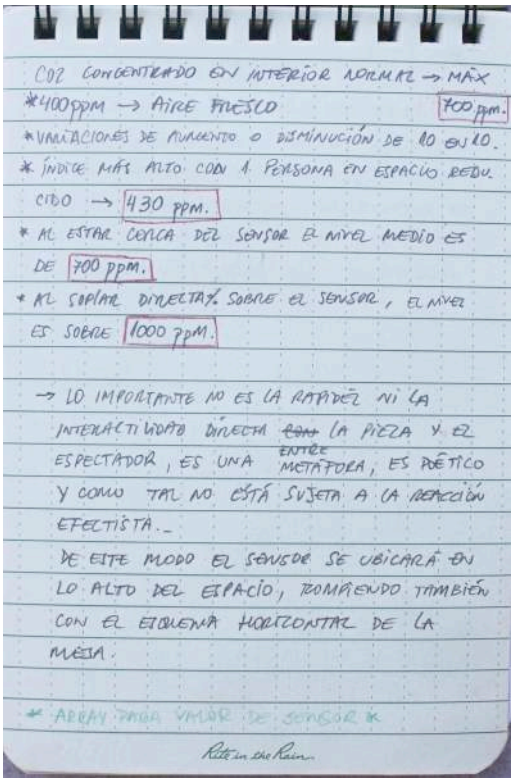


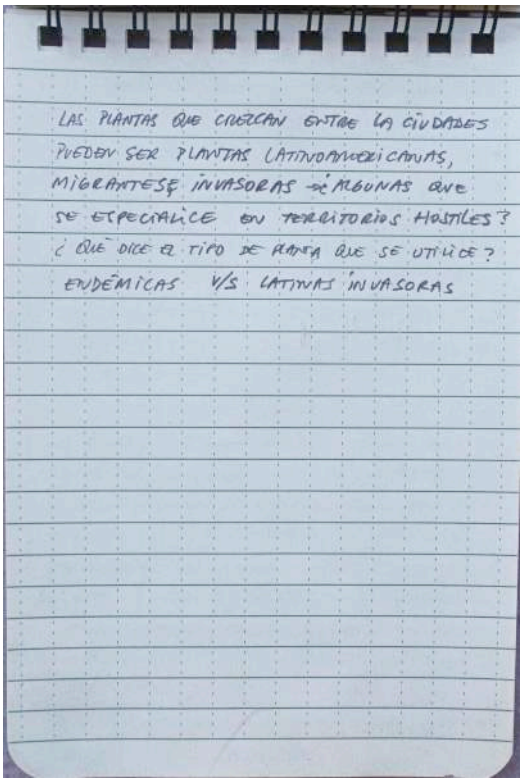
Bocetos de anatomía de plantas endémicas de la Comunidad Valenciana

Anotaciones para la planificación del prototipado II

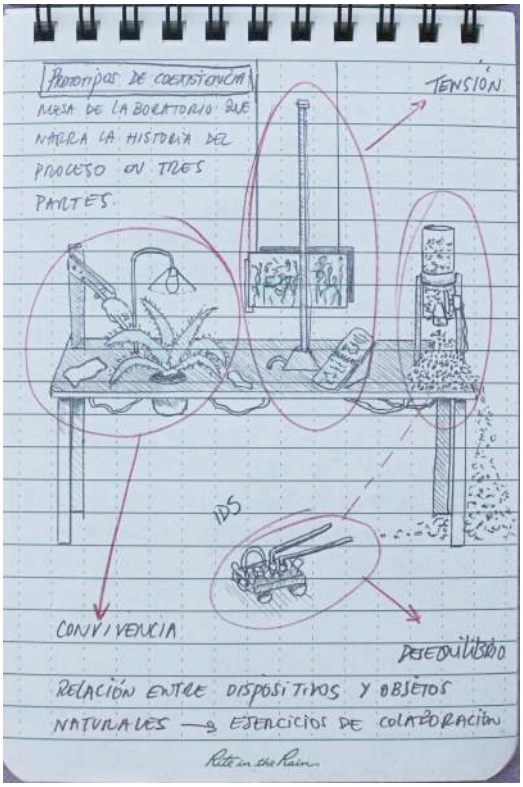


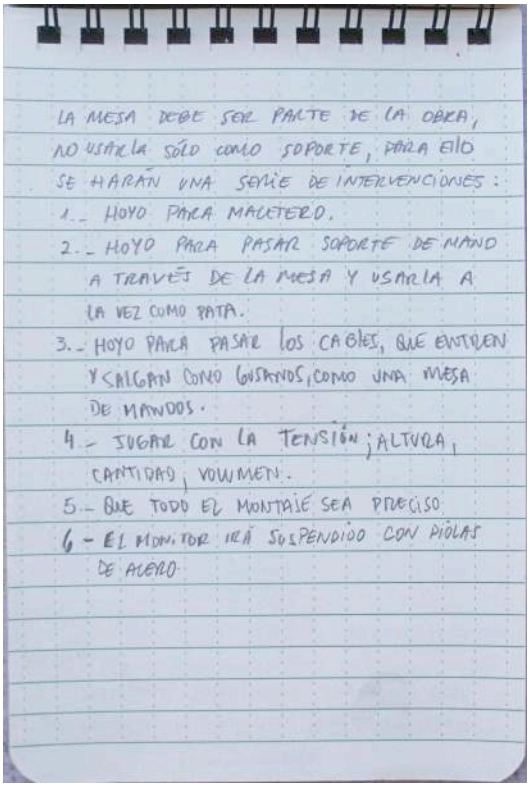


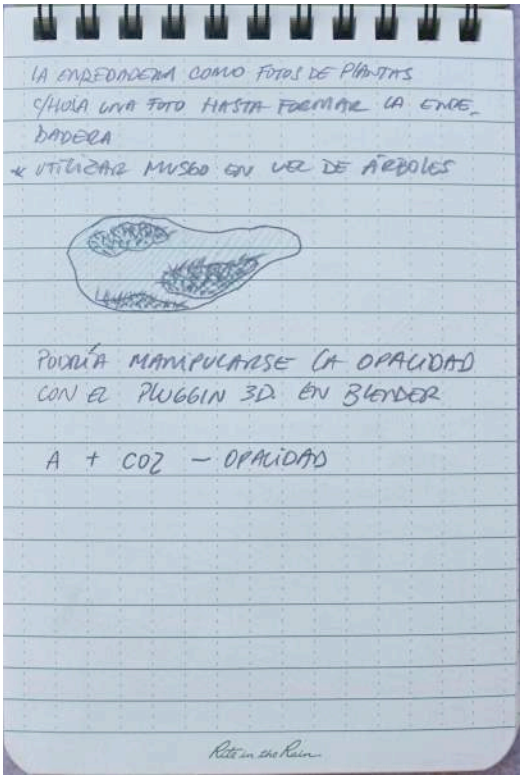


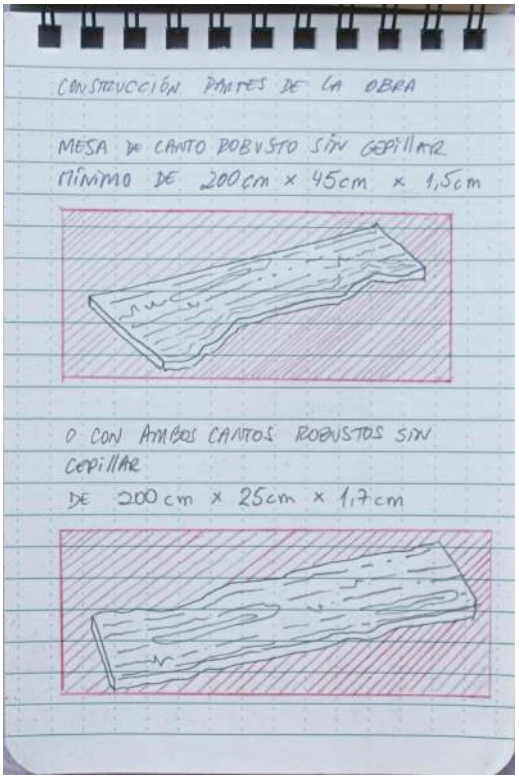


Propuestas para la construcción de la serie









Bocetos y esquemas técnicos de los prototipados