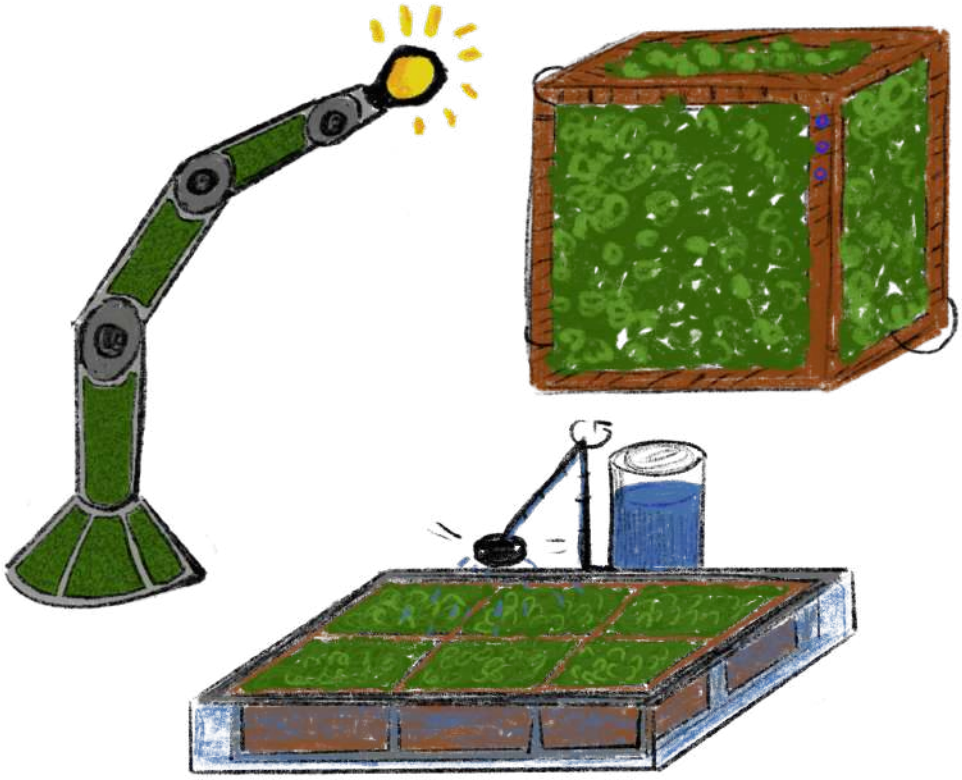
PLANT_ROBOT MUTUALISM

A starter guide



INTRODUCTION

In my graduation project, I explored mutualism between robot and plant. This mutualism is defined by an exchange of benefits, the robot provides water to the plant and in return the plant provides electricity. To do this, I used moss powered MFCs and an energy harvesting circuit. My objective was to learn about what it takes to build plant-robot interaction.

In this guide, I explain how to reproduce the set-up I have made. First going over the bryoMFCs setup and then the circuit.

Find the code and more documentation on github: https://github.com/Wrendraya/plant_robot_mutualism_thesis_doc

ELECTRICITY FROM PLANTS : BRYOMFCS

Bryophytia, more commonly called moss, Microbial Fuel Cell, or for short bryoMFC, are biochemical systems that transform chemical energy into electricity thanks to microbial activity. With MFC it is possible to extract energy from plants.

In this section, find out how to build 3 MFCS and connect them together to generate electricity. All the MFCs use the same materials and set-up. This completes the plant side of the plant-robot mutualism.

HOW DO THEY WORK ?

BryoMFCs transform organic matter into electricity using moss and bacteria. To generate electricity a particular setup is needed, composed of ;

- The cathode (positive electrode) placed in contact with the plant and the microbial activity. It's where the reduction takes places, which allows for the electrons to flow through the external circuit.

- The anode (negative electrode). It harvest the electrons created from the various chemical reaction of the microbial activity. It is connected by an external circuit to the cathode.

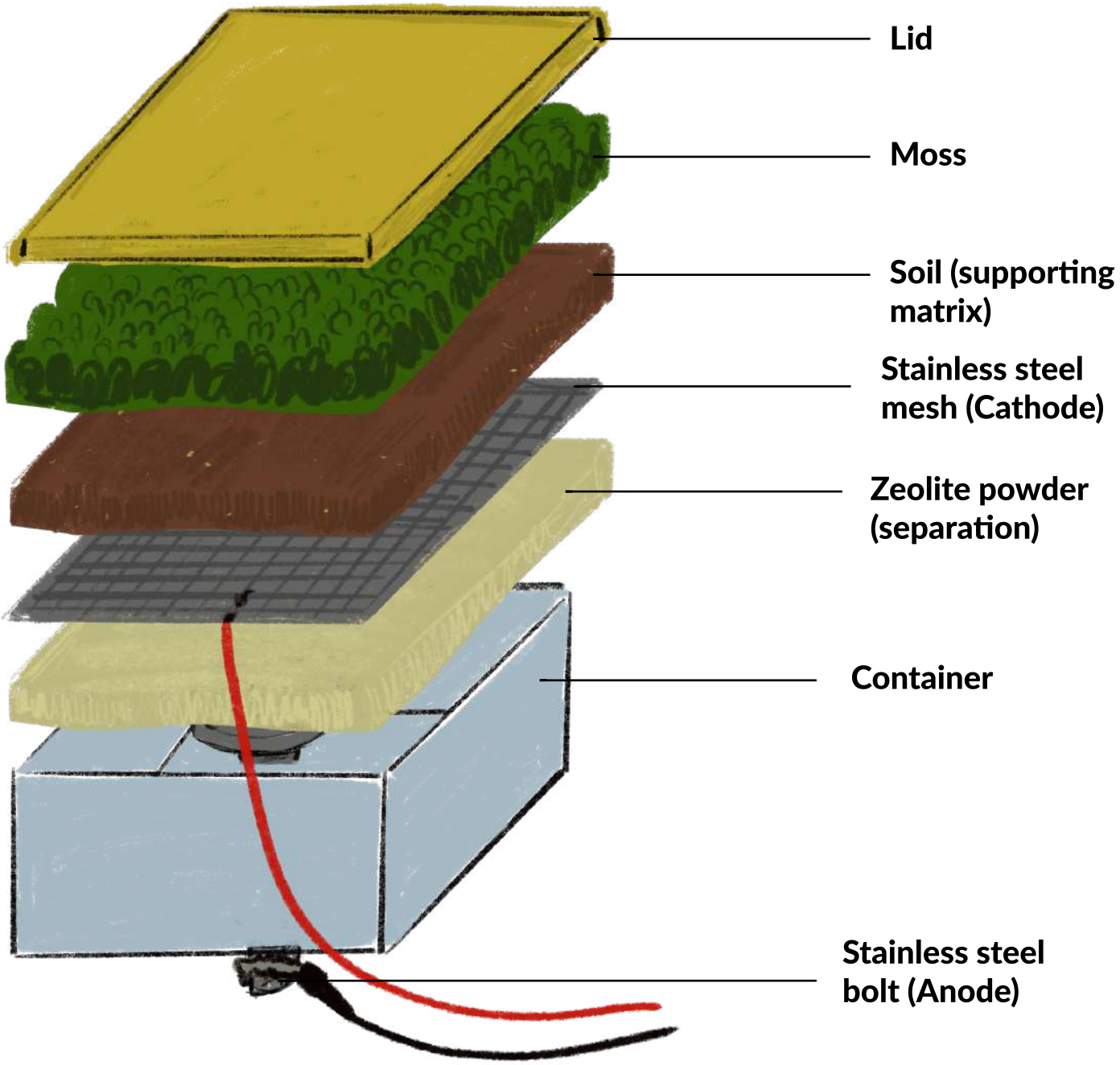
- The supporting matrix. Generally made out of soil or sediments, and where the cathode and the plant are buried.

- A separating membrane between the anode and the cathode.

For this set-up, we use stainless steel for the cathode and the anode. Soil for the supporting matrix and zeolite powder for separating the cathode and anode.

THE SET-UP

The bryoMFC set-up proposed in this paper is represented here.



MATERIAL LIST

Material needed for a 3 bryoMFC set-up.

Stainless steel mesh - 1 mm thickness

It could be thicker, 1mm is just the easiest to cut into shape.

Stainless/Galvanized steel bolts

Ideally M8x20mm.

A plastic container with a lid

The plastic should be thin since a hole needs to be made in it. A normal sized food container would do perfect.

3 wires

Like jumper wires.

4 cables with crocodile clips

Pot-ground

Not a lot is needed and soil can also be used instead of pot-ground.

200g of zeolite powder

Zeolite powder can be bought on various online website, especially one's focused on health and nutrition.

Freshly collected moss

Be sure to collect enough moss to cover the surface of your container.

A multi-meter

STEP BY STEP

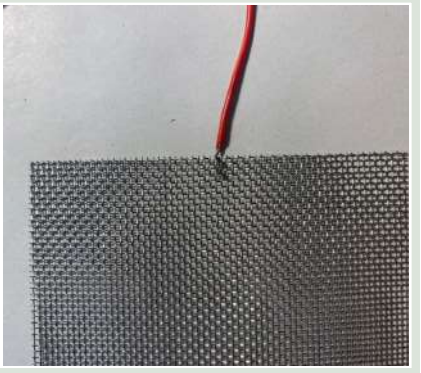
Step 1 : Gather the materials

Refer to the material list.

Step 2 : Making the Cathode

Start by cutting the stainless steel mesh to fit within the container you selected. Repeat the process 2 times. Then for each square of stainless steel mesh add a the wire by weaving it in.





Step 3 : Making the Anode

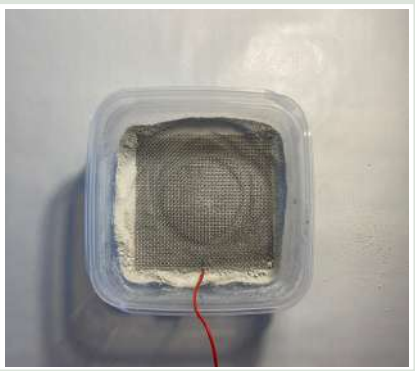
First, make a hole at the bottom of the container that is the size of your bolt. It should fit snug in it. You can use a soldering iron to make the hole easily. After the hole is done place the bolt through it with it's head towards the inside of the container. Repeat the process for remaining containers.



STEP BY STEP

Step 4 : Assemble

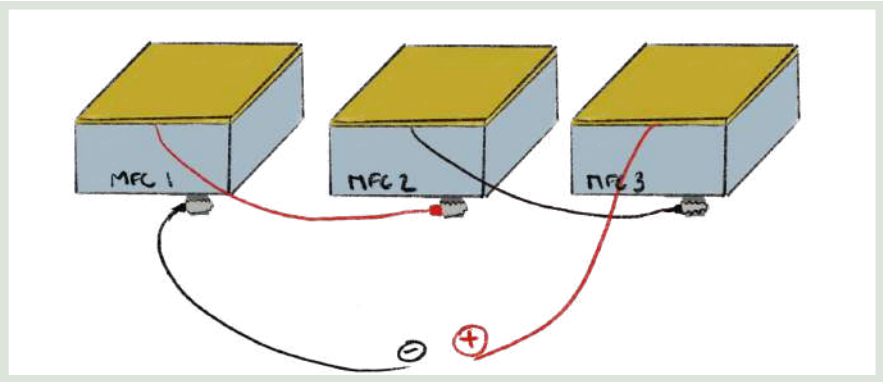
Take the container with the stainless steel bolt. Cover the stainless steel bolt with zeolite powder. Add till the top of the bolt is completely covered, the cathode should not be touching it. After that, add the cathode on top. Put the wire outside of the container before adding some pot-ground on top of it. After the pot-ground add the moss. Water it with a mist sprayer before closing with the lid. Repeat for the other 2.



STEP BY STEP

Step 5 : Connect in Series

To extract the most power from the MFCs, it's important to connect them with each other in series. Use the cables with crocodile heads to connect the stainless steel bolt to the cable of the other MFCs. Just as the drawing shows.



Step 6 : Measure the Voltage

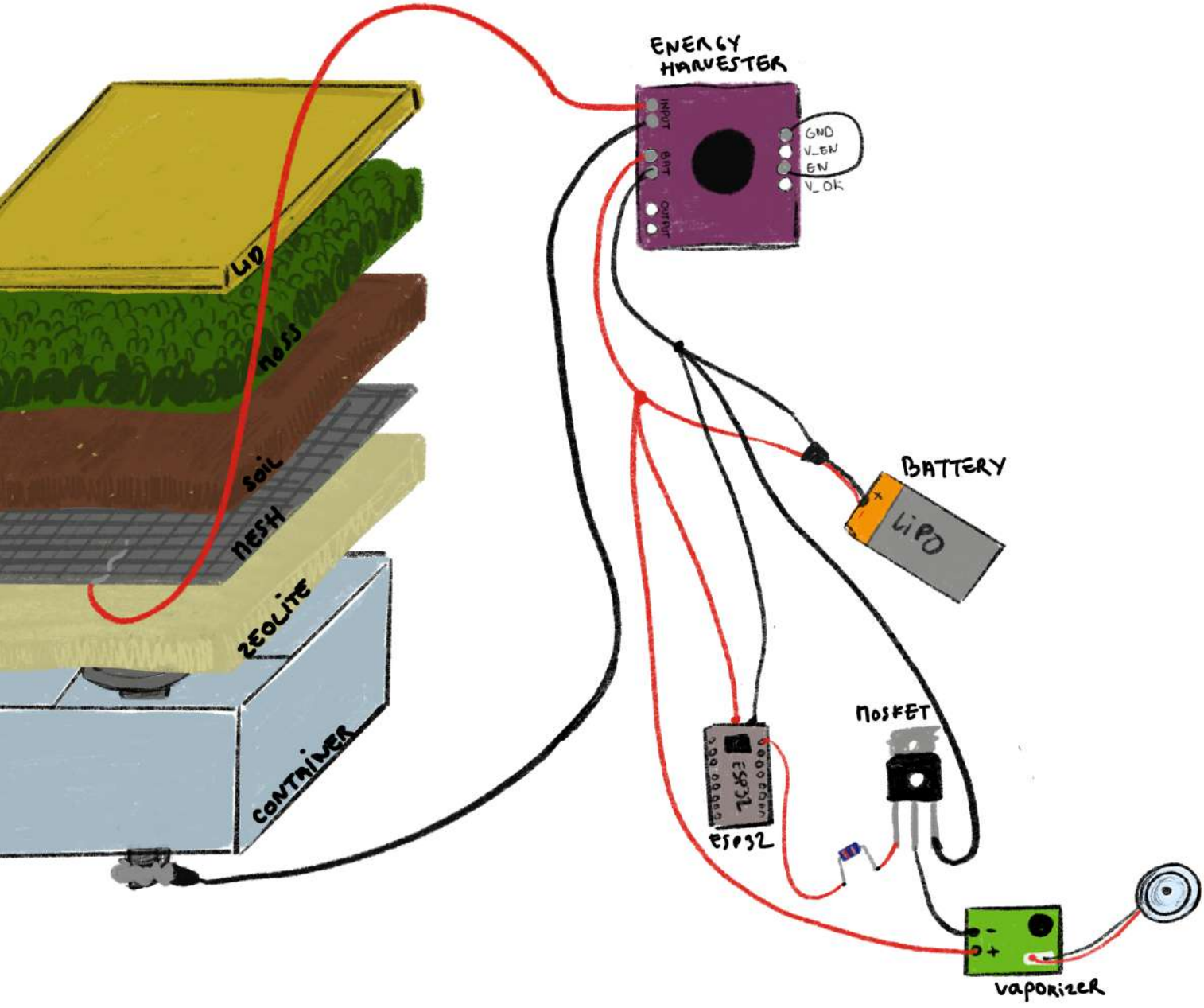
Now the bryoMFC set-up is complete, check the voltage using a multi-meter. The voltage measured might be very low, but it will slowly rise up in the next few days. Keep on checking the values.

ROBOTIC SIDE : ENERGY HARVESTING & MORE

In this section, I detail the robotic side of the mutualistic relationship. To realize the robotic side, I use an energy harvesting chip to collect the energy from the MFCs and charge a battery. The battery serves to power a micro-controller that turns on and off the vaporizer (the water system). Since the energy generated by the MFCs is very low, the battery takes a while to charge back. In order to not drain the battery, the micro-controller is programmed to go into a deep sleep mode. The deep sleep mode is programmed to last 12h, after which it wakes up and turns the micro-controller for 10s, before going back to sleep.

OVERVIEW

The overview of the circuit.



MATERIAL LIST

Material needed for the robotic side.

Ultrasone Mist Maker, OT3322-F41

A small vaporizer found online for around ~4 euros.

SEEED STUDIO XIAO ESP32S3

Includes a deep sleep mode and it can be found easily online for around ~9 euros.

N-channel MOSFET, IRLB8721

Wires

Energy Harvester, CJMCU-2557

Can be bought on AliExpress for 8euros.

Li-Po Battery, 3.7V, 250mAh

Soldering Iron Kit

The XIAO ESP32S3 can only use the battery power if soldered at the bottom. Other parts can be soldered, this is the only mandatory one.

Resistor 220ohm

THE CODE

Also on github : https://github.com/Wrendraya/plant_robot_mutualism_thesis_doc

```
/*
Simple Deep Sleep with Timer Wake Up
Part of the code by Pranav Cherukupalli <cherukupallip@gmail.com>
*/

const int transistorPin = 9; //actually pin D10 on the board//

int hours = 12;
int sleep_lenght = hours*3600;

#define uS_TO_S_FACTOR 1000000ULL  /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  sleep_lenght  /* Time ESP32 will go to sleep (in seconds) */

RTC_DATA_ATTR int bootCount = 0;

/*
Method to print the reason by which ESP32
has been awaken from sleep
*/
void print_wakeup_reason(){
  esp_sleep_wakeup_cause_t wakeup_reason;

  wakeup_reason = esp_sleep_get_wakeup_cause();

  switch(wakeup_reason)
  {
    case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Wakeup caused by external signal using RTC_IO"); break;
    case ESP_SLEEP_WAKEUP_EXT1 : Serial.println("Wakeup caused by external signal using RTC_CNTL"); break;
    case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Wakeup caused by timer"); break;
    case ESP_SLEEP_WAKEUP_TOUCHPAD : Serial.println("Wakeup caused by touchpad"); break;
    case ESP_SLEEP_WAKEUP_ULP : Serial.println("Wakeup caused by ULP program"); break;
    default : Serial.printf("Wakeup was not caused by deep sleep: %d\n",wakeup_reason); break;
  }
}

void setup(){
  pinMode (transistorPin, OUTPUT);
  digitalWrite(transistorPin, LOW);

  Serial.begin(115200);
  delay(1000); //Take some time to open up the Serial Monitor

  //Increment boot number and print it every reboot
  ++bootCount;
  Serial.println("Boot number: " + String(bootCount));

  //Print the wakeup reason for ESP32
  print_wakeup_reason();

  //Open/close mosfet
  delay(1000);
  digitalWrite(transistorPin, HIGH);
  Serial.println("Transistor High");
  delay(8000);
  digitalWrite(transistorPin, LOW);
  Serial.println("Transistor Low");
  delay(1000);
  /*
  First we configure the wake up source
  We set our ESP32 to wake up every TIME_TO_SLEEP seconds
  */
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  Serial.println("Setup ESP32 to sleep for every " + String(TIME_TO_SLEEP) +
  " Seconds");

  /*
  Next we decide what all peripherals to shut down/keep on
  By default, ESP32 will automatically power down the peripherals
  not needed by the wakeup source.The line below turns off all RTC peripherals in deep sleep.
  */

  Serial.println("Configured all RTC Peripherals to be powered down in sleep");

  /*
  Now that we have setup a wake cause and if needed setup the
  peripherals state in deep sleep, we can now start going to
  deep sleep.
  In the case that no wake up sources were provided but deep
  sleep was started, it will sleep forever unless hardware
  reset occurs.
  */
  Serial.println("Going to sleep now");
  Serial.flush();
  esp_deep_sleep_start();
  Serial.println("This will never be printed");
}

void loop(){
  //This is not going to be called
}
```

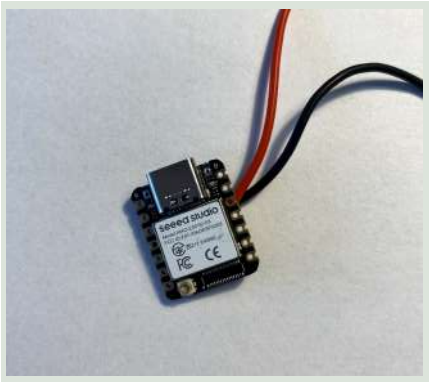
STEP BY STEP

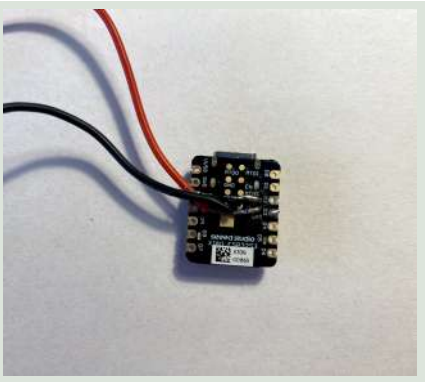
Step 1 : Gather the materials

Refer to the material list.

Step 2 : Solder the battery to the ESP32

The ESP32 can be powered by a 3.7V battery but for that it needs to be soldered on. In this case, only cables are soldered so that the battery can be interchanged easily in case of problems.





Step 3 : Code Micro-controller

The micro-controller does two things. First, it turns the vaporizer on using the MOSFET, and then enters deep sleep to conserve battery power. The code for the deep sleep uses a internal timer, since no external peripherals are used. The official XIAO ESP32S3 page offers the code for deep sleep with a timer, the only change concerns the time. The ESP32 in this case sleeps for 12h. Finally, the last addition is to code the MOSFET to be high and low. Upload the code (refer to the code page), using Arduino IDE, to your micro-controller.

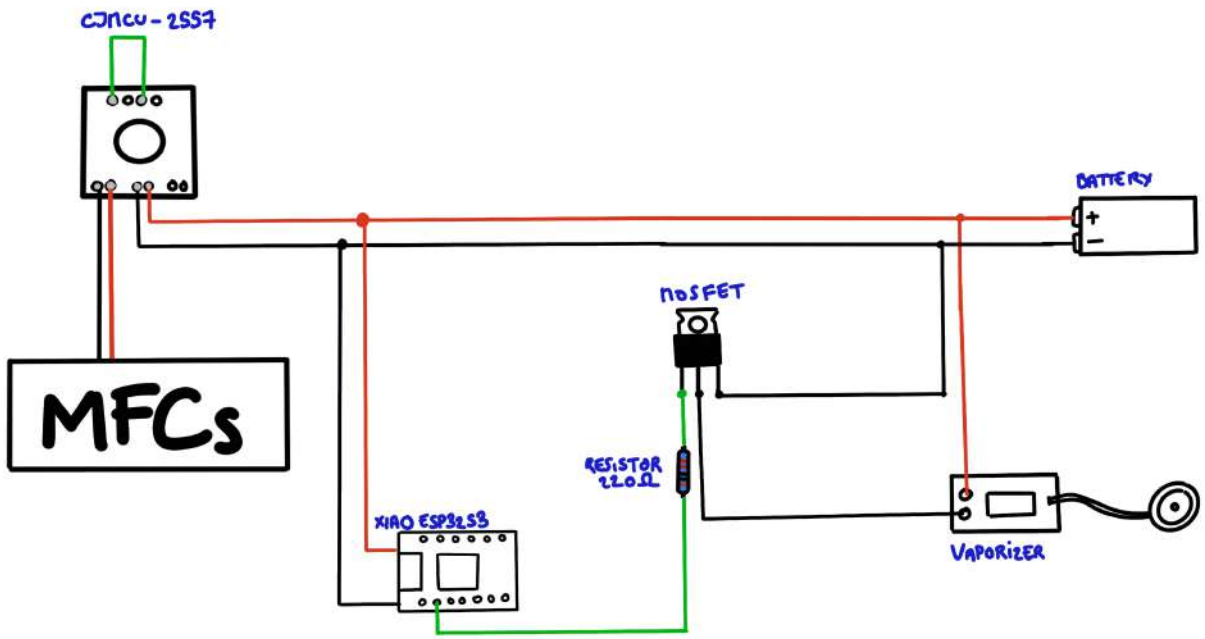
⚠️ To change the code of the ESP32 after being put in deep sleep, press the B button and keep it pressed down whilst you plug it in and upload the code.
⚠️ It happens that the micro-controller isn't seen by Arduino IDE. The best solution is then unplug the ESP32 and relaunch the IDE.
⚠️ The pin D10 of the board is actually seen as pin 9 in the code.

STEP BY STEP

Step 4 : The Circuit

Now, let's connect all the parts of the circuit together. Most of the parts aren't soldered together to easily change the build.

- Connect the GND pin and EN pin of the energy harvester (CJMCU-2557) to enable the chip.
- Connect the MFCs to the INPUT pin of the energy harvester.
- Connect the battery to the BAT pin of the energy harvester.
- Connect the micro-controller (XIAO ESP23S3) to the BAT pins of the energy harvester using the soldered cables.
- From the micro-controller pin D10 connect to the MOSFET pin with a 220 ohm resistor in-between.
- The connect the MOSFET to the negative side of the BAT pin, and as the negative pin of the Vaporizer.
- Finally, connect the vaporizer to the positive side of the BAT pin.



END WORD

This is only the beginning of what is possible in terms of plant-robot mutualism. They are other shapes to explore, types of interaction, improvements to be made on the circuit and MFCs.

Some question to develop the relationship further, could be :

- What other benefits could robots and plants exchange?
- Is there other ways to extract energy from plants?
- How can we improve the circuit, especially when it comes to the watering system?
- Can we replace the micro-controller with a simpler alternative to turn on/off the vaporizer?