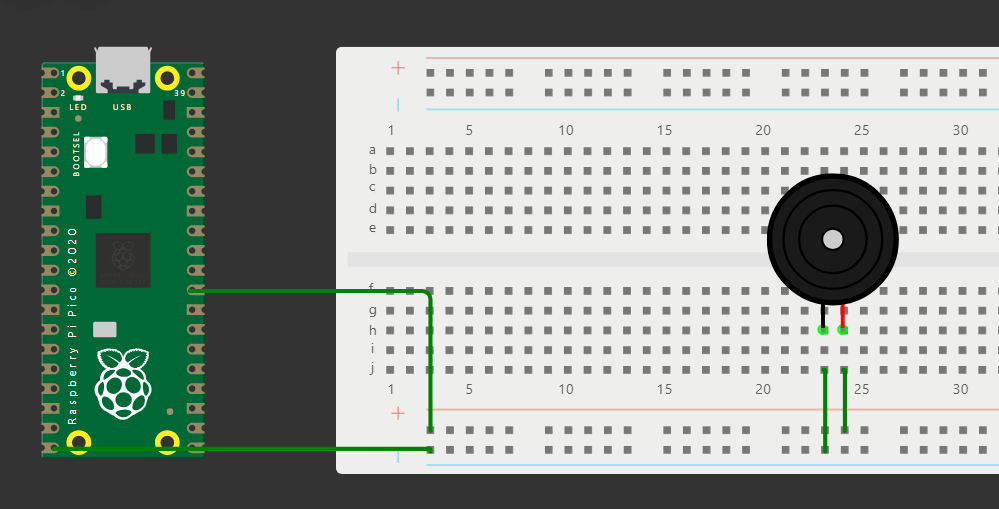
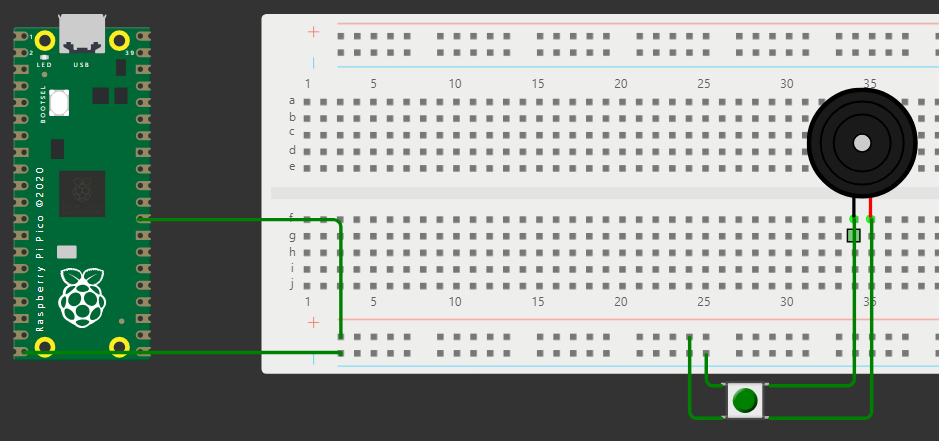
**PRACTICE 3: MODERN IOT**

**1. a buzzer**

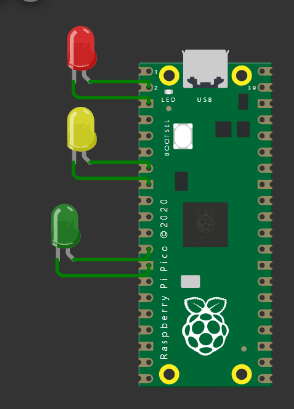
1.1 Connecting an active buzzer



1.2 Make Bell



1.3 Making traffic lights



Code:

import machine

import time

# Define the GPIO pins for the traffic light LEDs

RED\_LED\_PIN=1 #GP0

YELLOW\_LED\_PIN=5 #GP1

GREEN\_LED\_PIN=9 #GP2

#Intialize the GPIO pins for the LEDs

red\_led = machine.Pin(RED\_LED\_PIN,machine.Pin.OUT)

yellow\_led = machine.Pin(YELLOW\_LED\_PIN,machine.Pin.OUT)

green\_led = machine.Pin(GREEN\_LED\_PIN,machine.Pin.OUT)

# Function to turn on the specified LED and turn off the others

def turn\_on\_led(led\_pin):

red\_led.value(led\_pin == RED\_LED\_PIN)

yellow\_led.value(led\_pin==YELLOW\_LED\_PIN)

green\_led.value(led\_pin==GREEN\_LED\_PIN)

# Traffic light sequence

while True:

# Red light

turn\_on\_led(RED\_LED\_PIN)

time.sleep(5) # Red light for 5 seconds

# Red and yellow lights (transition)

turn\_on\_led(YELLOW\_LED\_PIN)

time.sleep(2) # Red and yellow lightS for 2 seconds

# Green light

turn\_on\_led(GREEN\_LED\_PIN)

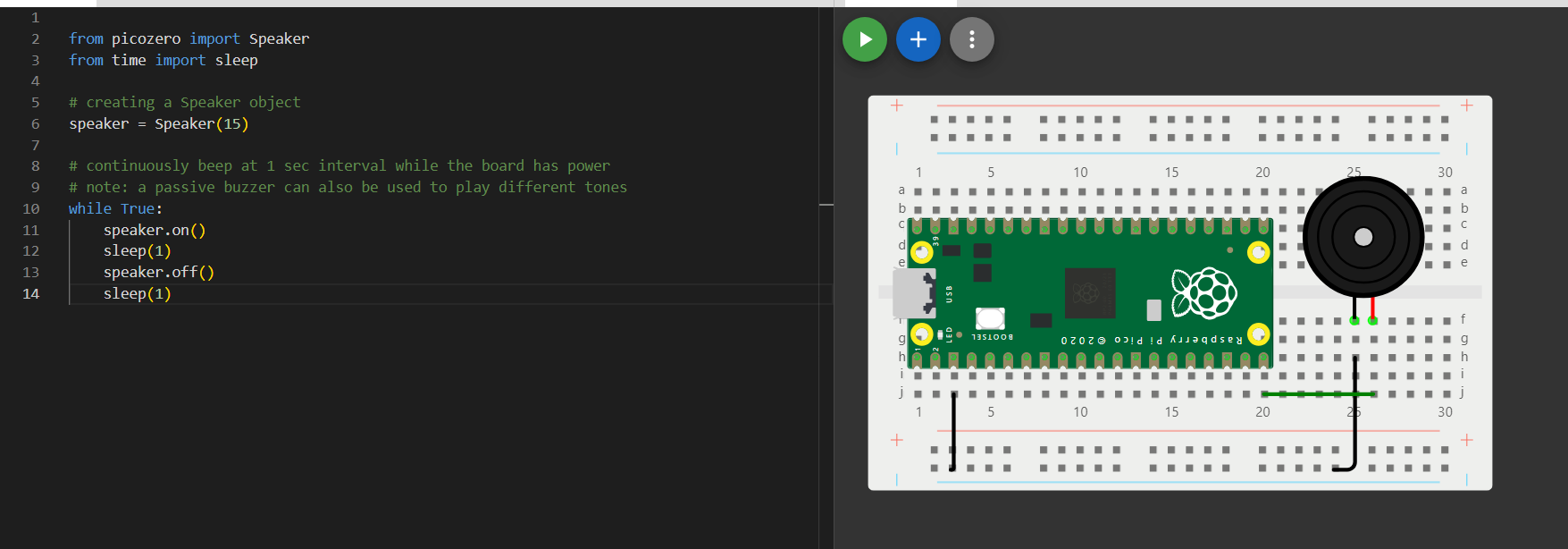
time.sleep(5) # Green light for 5 seconds

# Yellow light

turn\_on\_led(YELLOW\_LED\_PIN)

time.sleep(2) # Yellow light for 2 seconds

1.4 Passive Buzzers



Code:

from picozero import Speaker

from time import sleep

# creating a Speaker object

speaker = Speaker(15)

# continuously beep at 1 sec interval while the board has power

# note: a passive buzzer can also be used to play different tones

while True:

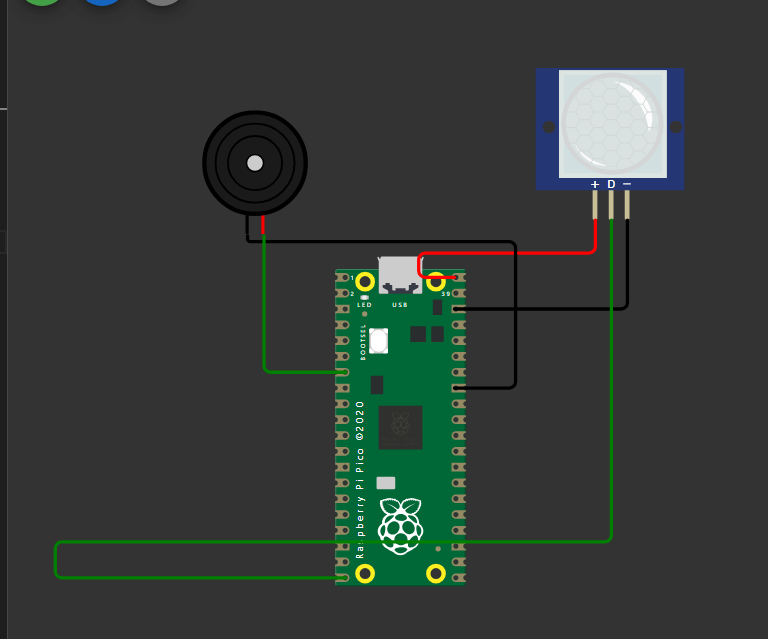
speaker.on()

sleep(1)

speaker.off()

sleep(1)

**2. Using a PIR sensor**

****

**Code:**

void setup() {

pinMode(15, INPUT);

Serial1.begin(115200);

Serial1.println("Hello, Raspberry Pi Pico!");

}

void loop() {

int pir=digitalRead(15);

if(pir == HIGH)

{

Serial1.println("MOVIMIENTO DETECTADO");

delay(500);

}

else

{

Serial1.println("NO MOVIMIENTO DETECTADO");

delay(500);

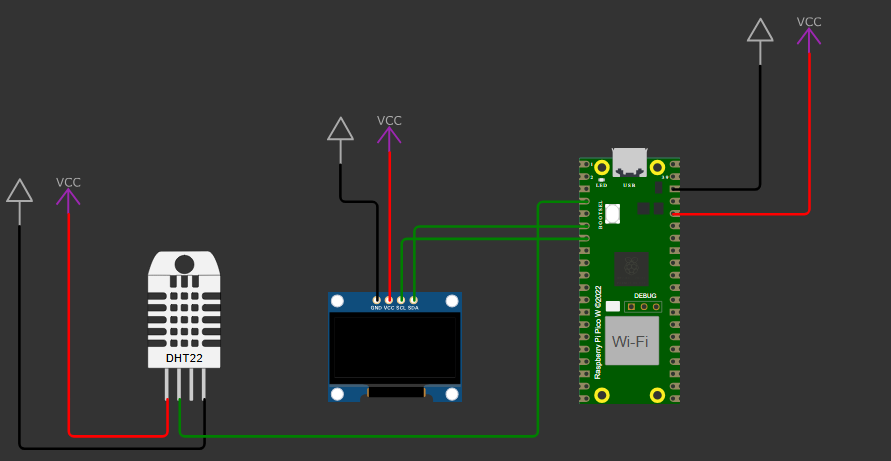
}

}

**3. OLED, LCD display**

3.1 Use a temperature sensor and display on OLED the temperature and

humidity data

****

code:

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

#define OLED\_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

#define DHTPIN 2 // Pin which is connected to the DHT sensor

#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(115200);

dht.begin();

// SSD1306\_SWITCHCAPVCC = generate display voltage from 3.3V internally

if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64

Serial.println(F("SSD1306 allocation failed"));

for(;;); // Don't proceed, loop forever

}

display.display();

delay(2000); // Pause for 2 seconds

// Clear the buffer

display.clearDisplay();

}

void loop() {

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

// Read temperature as Celsius (the default)

float t = dht.readTemperature();

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t)) {

Serial.println(F("Failed to read from DHT sensor!"));

return;

}

// Clear the display buffer

display.clearDisplay();

// Display temperature

display.setTextSize(1);

display.setTextColor(SSD1306\_WHITE);

display.setCursor(0,0);

display.print(F("Temp: "));

display.print(t);

display.println(F(" C"));

// Display humidity

display.setCursor(0,10);

display.print(F("Humidity: "));

display.print(h);

display.println(F(" %"));

// Display the buffer

display.display();

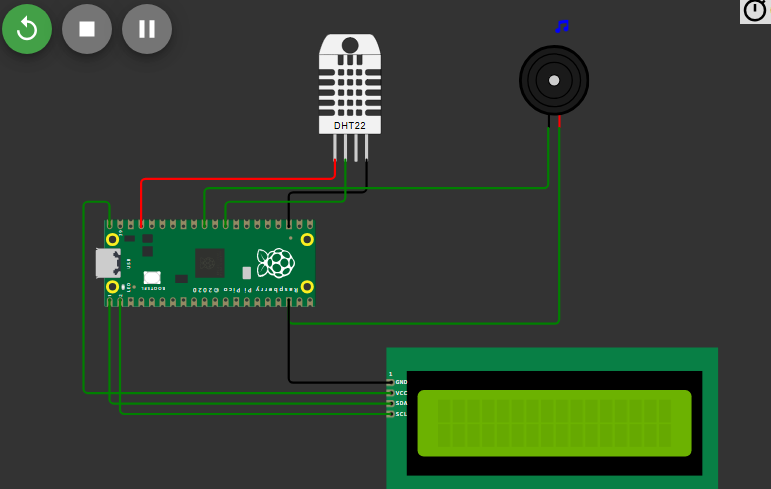
// Wait a few seconds between measurements.

delay(2000);

}

3.2 Use a temperature sensor and display on LCD the temperature and humidity

data each minute



Code:

from machine import Pin,PWM

from time import sleep

from dht import DHT22

from machine import I2C, Pin

from pico\_i2c\_lcd import I2cLcd

def buzzerFunc():

buzzer= PWM(Pin(26))

buzzer.freq(500)

buzzer.duty\_u16(1000)

sleep(1)

buzzer.duty\_u16(0)

# creating a DHT object

# change DHT22 to DHT11 if DHT11 is used

dht = DHT22(Pin(22))

i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)

# getting I2C address

I2C\_ADDR = i2c.scan()[0]

# creating an LCD object using the I2C address and specifying number of rows and columns in the LCD

# LCD number of rows = 2, number of columns = 16

lcd = I2cLcd(i2c, I2C\_ADDR, 2, 16)

# continuously get sensor readings while the board has power

while True:

# getting sensor readings

dht.measure()

temp = dht.temperature()

hum = dht.humidity()

# displaying values to the console

print(f"Temperature: {temp}°C Humidity: {hum}% ")

lcd.putstr(f" Temp: {temp}°C Humty: {hum}% ")

sleep(5) # "Hello world!" text would be displayed for 5 secs

lcd.clear()

sleep(1) # clear the text for 1 sec then print the text again

# format method or string concatenation may also be used

#print("Temperature: {}°C Humidity: {:.0f}% ".format(temp, hum))

#print("Temperature: " + str(temp) + "°C" + " Humidity: " + str(hum) + "%")

# delay of 2 secs because DHT22 takes a reading once every 2 secs

sleep(2)

if(temp > 20 and hum > 30):

buzzerFunc()

else:

None