Anexos

Código principal ejecutado desde el ordenador

```
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#::::::::::::::: #woman critical interface by Silvia Binda Heiserova
 :::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#::::::::::::::::::::: MAIN CODE TO BE EXECUTED FROM THE PC
 :::::::::::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::

### THIS CODE DOES:
# 1. Parsing data from instagram (hashtags related to the hashtag "#woman"
 and total count of publications with the hashtag "#woman")
# 2. Sending messages via wifi to esp32(1), esp32(2) and internally to a
 .py code executed on this PC
# 3. Receiving messages via from esp32(1) (median value measured by US
 sensor)
# 4. Converting text to speech
# 5. Creating a backup database of hashtags in a separate .py file

import ssl
import json # library for parsing json data
import unicodedata as ud # to recognize latin characters
import traceback # library for printing errors
from urllib.request import urlopen # library for url reading
from socket import * # library for network communication
from requests.exceptions import HTTPError # library for exceptions
import string # library for converting to string
from time import sleep
import pyttsx3 # library for converting text to speech
import datetime # to print the current date and time
import random # library for random values
from _thread import start_new_thread # library for threads
from backup_updated_hashtags_count import backup_hashtags, backup_count #
 importing our own .py file we created to automatically store parsed
 hashtags and total count (this file has to be in the same folder)

s = socket(AF_INET, SOCK_DGRAM) # for network UDP communication
# a pair (host, port) is used for the AF_INET address family, where host is
 a string representing either a hostname in internet domain notation like
 'daring.cwi.nl' or an IPv4 address like '100.50.200.5', and port is an
 integer
# SOCK_DGRAM is for UDP socket communication

# following 3 lines we un/comment when we are testing the code without
 actual HW connection of other components:
esp32_1 = ('127.0.0.1', 8347) #TESTING !!
esp32_2 = ('127.0.0.1', 8347) #TESTING !!
s.bind(('127.0.0.1', 8345)) #TESTING !!

#esp32_2 = ('192.168.1.103', 8345) # we add the ip adress of esp32(2) and
 the port
#esp32_1 = ('192.168.1.102', 8345) # we add the ip adress of esp32(1) and
 the port
```

```
iterm_konzola = ('127.0.0.1', 8346) # we add the local ip adress of the
 computer to send data to be transformed into binary code within a code
 executed in iterm (finally visualizaed on CRT TV)
#s.bind(('192.168.1.101', 8345)) # we add the ip adress of this computer
 (the one we want to send the data to other destinations like
 microcontrollers), and the number of the assigned port

url = "https://www.instagram.com/explore/tags/woman/?__a=1" # we define the
 url adress from where we will scrape the data (in our case the data is the
 total number of publications on instagram with the hashtag #woman, as well
 as related hashtags that appear in publications with the hashtag #woman)

count = backup_count # we set an initial int value of the total number of
 publications with the hashtag #woman
time_to_sleep = 2 # sets time to sleep for the "fake count" of total
 publications number (when exception occurs and we enter in "panic mode")
new_hashtags_result =["#woman", "#criticalinterface", "#interface",
 "#women", "#test", "interfacetest"] # we create an array of strings where
 the new incoming hashtags from instagram will be stored
hashtag_index = 0 # we set an initial int value for the index of the
 hashtags in the array
cache_history = [] # to store last 5 hashtags scraped from instagram (to
 avoid repetition of the hashtags)
distance = 100 # set initial value for distance of the ultrasound sensor,
 it is > 99 because we want the voice to start in reverse
# initial settings for text to speech conversion:
engine = pyttsx3.init() # object creation for tts
rate = engine.getProperty('rate') # to set rate of the speaking voice
engine.setProperty('rate', 101)
volume = engine.getProperty('volume') # to set the volume level of the
 speaking voice (min=0 and max=1)
engine.setProperty('volume', 1)
voices = engine.getProperty('voices') # to set the type of the voice
 speaking
engine.setProperty('voice', voices[0].id) # changing index changes voices
 (0 for male, 1 for female...in total there are 18 differen voices)

# function for parsing the total number of publications with the hashtag
 #woman from a json:
def parser_count():
    global json_data
    # we use the key-value pairs of the JSON file to create a Python
     dictionary that we can use in our program to read the data:
    return
     int(json_data['graphql']['hashtag']['edge_hashtag_to_media']['count'])
     # with this path we get the int value of total publications with the
     hashtag #woman
    # for documentation on parsing json data see:
     https://www.geeksforgeeks.org/convert-json-to-dictionary-in-python/,
     https://www.freecodecamp
     .org/news/python-read-json-file-how-to-load-json-from-a-file-and-parse-
     dumps/

# main function for handling the data from instagram (will be called in the
 first thread)
```

```
def instagram_data_handler():
    global count, time_to_sleep, new_hashtags_result, hashtag_index,
     json_data, backup_updated_hashtags
    # we create a boolean variable for 2 different states:
    # False is when everything works well and we can scrape RTD from
     instagram url
    # True is when we need to "fake" the rise of numbers
    fake_numbers_state = False
    fake_counter = 0 # we create an int variable to count the amount of
     fake additions to the total count
    previous_state = False # to check the previous state of fake counter
     and compare it with the current state

    # we create an array of 30 items where the rise (base on RTD) of total
     number of publications will be stored for the case we have to enter
     the "panic mode":
    difference_array = [1, 1, 1, 3, 5, 2, 1, 5, 3, -1, 2, -1, 5, 0, 3, 2,
     1, 1, 4, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 0]
    # just a note/hint silvia: difference_array = [2]*30 ### means 30 x "2"
     items in array

    while True:
        if not fake_numbers_state: # means if fake_numbers_state = False
         (everything is working ok)
            try:
                ssl_context = ssl._create_unverified_context()
                response = urlopen(url, timeout=5, context=ssl_context) #
                 opens our instagram url with the data we need
                # we have to add a timeout limit (in seconds) because by
                 default its timeout is None (in other words, urlopen would
                 never timed out, which was causing us problems), for
                 details see:
                 https://docs.python.org/3/library/socket
                 .html?highlight=socket#socket.getdefaulttimeout
                json_data = json.loads(response.read()) # loads the url as
                 json
                old_count = count # we create a variable to compare the
                 previous count with the actual count
                count = parser_count() # parser_count is our function we
                 created to get the total number of publications with
                 hashtag #woman, returns an int
                new_hashtags_result = get_latest_related_hashtags() # we
                 fill the variable with an array of newly parsed hashtags
                hashtag_index = 0 # we set the hashtag_index back to 0, so
                 it starts reading and showing the hashtags from the
                 beginning of the array
                if (count-old_count) >= -5 and (count-old_count)<5: # if
                 the difference between the actual and the previous count
                 is equal or higher than -5, and it is not bigger than 5
                 (would be some error), then add to the array of number the
                 difference by which the count rises
                    difference_array.append(count-old_count) # adds new
                     number to the array (we excluded minus values but
                     included 0)
                    del difference_array[0] # deletes the first number
                     (item) from the array
                    print(difference_array)
                previous_state = fake_numbers_state # previous state will
                 be False

            except Exception as e: # exception for when something goes
             wrong, we will use our "fake" numbers
                traceback.print_exc() # prints the error details
                fake_numbers_state = True # change the state to True (=
                 activate panic mode)
                if not previous_state: # if 2 continous fake states occur,
                 we want to continue the hashtag index, otherwise we want
                 to start from 0
                    hashtag_index = 0
                previous_state = fake_numbers_state # we set the previous
                 state to True
                fake_counter=0 # we set the fake counter to zero to start
                 counting again up to 30
                print("Exception ocurred. Switching to fake number state",
                 e)
                continue

            finally: # here we put the block of code we always want to
             execute:
                if len(new_hashtags_result) < 7 and len(backup_hashtags) >=
                 100: # if the array has less than 7 items, it means it
                 could not parse any hashtags (because the initial array
                 has 6 items)
                    new_hashtags_result = backup_hashtags[:] # we will use
                     our long predefind array of hashtags to be shown and
                     spoken
                    fake_numbers_state = True # we will activate the fake
                     numbers state
                if len(new_hashtags_result) < 7 and len(backup_hashtags) <
                 100: # if the array has less than 7 items, it means it
                 could not parse any hashtags (because the initial array
                 has 6 items)
                    new_hashtags_result = preset_backup_hashtags[:] # we
                     will use our long predefind array of hashtags to be
                     shown and spoken
                    fake_numbers_state = True # we will activate the fake
                     numbers state
                if not previous_state: # if 2 continous fake states occur,
                 we want to continue the hashtag index, otherwise we want
                 to start from 0
                    hashtag_index = 0
                previous_state = fake_numbers_state
                fake_counter = 0 # we set the fake counter to zero to start
                 counting again up to 30
                print('total number of publications:', count,
                 "fake_numbers_state", fake_numbers_state)
        else: # if fake_numbers_state is True and we are in "panic mode",
         do the following:
            try:
```

```
                count+=difference_array[fake_counter%len(difference_array)]
                 # we will continousely add 1 number from the difference
                 array o the count, the index of the number from the array
                 will get back to 0 after going through the whole array
                 (thanks to using fake_counter%len(difference_array))
                sleep(time_to_sleep) # sets time to sleep for the "fake
                 count" of total publications number, means how often will
                 the count update when in panic mode
                fake_counter+=1 # we augment our fake counter by 1 each
                 time it creates a new fake count
                print ("fake counter nr:", fake_counter)
            except Exception:
                traceback.print_exc() # prints the error details

        if fake_counter == 30: # if fake counter reaches the value 30 ->
         after 30 "fake" updates of count, try again to access RTD (starts
         again this while True cycle)
            fake_numbers_state = False

        # send data to esp32(2) converting it into a string and formatting
         it with 2 spaces between thousands:
        s.sendto ((str(count)[:2]+ " " + str(count)[2:5]+ " " +
         str(count)[5:8]).encode(), esp32_2)
        print(fake_numbers_state, count)

# function for filtering the hashtag to only those with ABC and abc letter
 (no numbers, no emoticons, no other signs...s)
# we have replaced the str.isalpha() with this function because
 str.isalpha() in python3 includes also different signs... and we only want
 ABC and abc letters to be included in our hashtags:
def only_letters(tested_string):
    for letter in tested_string:
        if letter not in
         "#abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVXYZ": # we have
         to include the "#" sign as well because is it part of the string
         we are checking (in: for x in process3...)
            return False
    return True

# function for getting hashtags related to the hashtag #woman:
def get_latest_related_hashtags():
    global json_data, posts, other_hashtags
    posts =
     json_data['graphql']['hashtag']['edge_hashtag_to_media']['edges']
    other_hashtags = []
    # we have to filter the parsed json data to get only the hashtags we
     want:
    # because in the json sometimes the hashtags are not divided by empty
     space, like #woman#art#girl, we have to divide them manually:
    for p in posts:
        try:
            text = '
             '.join(p['node']['edge_media_to_caption']['edges'][0]['node']
             ['text'].split()) # split is default by space
```

```
                process = [x for x in text.split(' ') if len(x)>2 and x[0] ==
                 '#'] # we take only those strings that start with "#"
                process2 = []
                for pr in process: # for cases when the next hashtag is on new
                 line
                    if len(pr.split('\n')) > 1:
                        process2 += [x for x in pr.split('\n') if len(x) > 2]
                    else:
                        process2.append(pr)
                process3 = []
                for pr in process2: # filter for if there are more hashtags
                 signs # in one continous string
                    if len(pr.split('#')) > 2:
                        process3 += ['#'+x for x in pr.split('#') if len(x) > 2]
                    else:
                        if len(pr) > 3: # we include only thos hashtags which
                         have at least 4 elements (min. 3 letters)
                            process3.append(pr)
                # we limit the the hashtag to ABC and abc only, and the length
                 of hashtag to maximum 10 letters
                #process4 = [x for x in process3 if only_roman_chars(x) and
                 len(x)<11]
                process4 = []
                for x in process3:
                    if only_letters(x) and len(x)<11:
                        process4.append(x)
                other_hashtags += process4

            except:
                continue
        print("hashtag array length:", len(other_hashtags))
        #print(other_hashtags)
        return other_hashtags # this function returns an array of relates
         hashtags

    # function for determing 1 sole hashtag that should be shown on oled
     display and spoken by tts:
    def get_next_hashtag():
        global hashtag_index, cache_history
        if hashtag_index >= len(new_hashtags_result):
            hashtag_index = 0 # we set the index back to zero in case the
             hashtag index should come to the end of the array
        counter = 0
        next_hashtag = new_hashtags_result [hashtag_index] # next hashtag to be
         shown and spoken is the next from the new hashtag result array
        hashtag_index+=1
        # we include a cache_history where we compare the last 5 hashtags (so
         we will not to repeat the same hashtags among the last 5)
        while next_hashtag in cache_history:  # while the next hashtag should
         be in the cache history, we skip it and go to the next one
            print(next_hashtag, cache_history)
            if hashtag_index >= len(new_hashtags_result):
                hashtag_index = 0
            next_hashtag = new_hashtags_result [hashtag_index]
            hashtag_index+=1
            counter+=1
            if counter > len(new_hashtags_result): # if we should pass all the
             hashtags from all hashtags array, we clear the cache history and
             start filling it again
                cache_history = []
                break # we break out of the loop

        if hashtag_index >= len(new_hashtags_result):
            hashtag_index = 0

        cache_history.append(next_hashtag) # we append the next hashtag to the
         cache history and if it has more than 5 hashtags in history, we delete
         the first one
        if len(cache_history)>5:
            del cache_history[0]

        return next_hashtag

    # function for receiving the distance measured by the ultrasound sensor,
     will be called in the second thread:
    def recvdatasensor():
        global distance
        while True:
            distance, dir = s.recvfrom(1024) # we recieve the distance value
             from the us sensor connected to esp32_1
            distance = float(distance.decode("ascii")) # change the received
             data to a float
            print(distance)

    # function for converting the hashtag to speach and for showing it on oled
     display:
    def hashtag_show_speak():
        global distance
        while True:
            try:
                hashtag_to_show = get_next_hashtag() # the hashtag to show on
                 oled display equals the string returned by the function called
                 here
                if len(hashtag_to_show) == 1: # to exclude cases when the
                 hashtag only consists from one item - especially the hashtag
                 "#"
                    hashtag_to_show = "#woman" # in that case it will show
                     always "#woman"
                hashtag_to_say = hashtag_to_show[1:] # the hashtag to be red by
                 tts will be the same as the one showed on display but without
                 the first element "#"
                # for converting text to speech:
                engine.setProperty('voice', voices[0].id) # sets male voice as
                 default
                # if the distance from sensor is higher than X cm reads the
                 hasthags in reverse order and changes to female voice,
                 otherwise, if it is less then X cm, reads in regular order
                 with male voice:
                if distance < 100:
```

```
                engine.setProperty('voice', voices[10].id) # changes to
                 female voice
                engine.say(hashtag_to_say) # says the hashtag
            else:
                engine.say(hashtag_to_say [::-1]) # if the distance from
                 the sensor is 100 cm or more, the order of the letters
                 will be reversed (from to by one, minus is for starting on
                 the last item)
            s.sendto(hashtag_to_show.encode(errors="ignore"), esp32_1) #
             sending text to ESP32(1) (to be shown on oled display)
            s.sendto(hashtag_to_say.encode(errors="ignore"), iterm_konzola)
             # sending text to iTerm (to be shown on CRT TV as binary code)
            print(hashtag_to_show)
            engine.runAndWait()
            #sleep(0.1) # !!! note: try changing sleep (on /off) if not
             working correctly

        # if any exception occures, print the error and continue:
        except Exception as e:
            print('Exception ocurred', e)
            traceback.print_exc() # prints the error details
            continue

# function for storing hashtags and count in a separate .py file we are
 importing:
def create_backup():
    global new_hashtags_result, backup_hashtags, other_hashtags
    while True:
        try:
            if len (other_hashtags) > 500 and backup_hashtags !=
             other_hashtags: # we update the backup only if the array of
             hashtags has more than 500 strings and only if it is different
             to the previous backup
                now = datetime.datetime.now() # to show the date and time
                 of the update
                file =
                 '/Users/admin/Desktop/FINAL_CODES_WCI_18_04_2022/backup_upd
                 ated_hashtags_count.py' # path to the .py file we are
                 updating
                with open (file, 'w') as f: # 'w' is to overwrite, 'a' is
                 for append
                    f.write('backup_hashtags = {}'.format(other_hashtags) +
                     '\n''backup_count= {}'.format(parser_count()) +
                     '\n\n#Updated on: {}'.format(now.strftime("%Y-%m-%d
                     %H:%M:%S")))
                print((now.strftime("%Y-%m-%d %H:%M:%S")) + " Updated the
                 backup file") # to know the exact date and time when the
                 database was updated
                backup_hashtags = other_hashtags
                sleep(300) # we update this array every 5 minutes

        except:
            continue
```

```
# we define an array of strings (hashtags) which will serve as a backup in
 case that the other backup file should for some reason be empty or the
 connection to instagram will be lost or otherwise inhibited, so that the
 interface can continue its functioning, simulating the real time data
 (this backup array of strings was downloaded from publications with the
 hashtag #woman from instagram on April 2nd, 2022 ):

preset_backup_hashtags = ['#popart', '#art', '#artista', '#fridakahlo',
 '#streetart', '#love', '#beauty', '#bloggerstyle', '#bloggerfashion',
 '#blogger', '#bloggers', '#blog', '#influencer', '#azeriblogers',
 '#azeriblogger', '#azeribloggers', '#azeriqizlar', '#kadin', '#kadinlar',
 '#qadinlar', '#qadin', '#qadinfm', '#womanlook', '#womanstyle',
 '#styleblogger', '#style', '#look', '#womancollection', '#girl', '#kadin',
 '#kadinlar', '#woman_rich', '#shein', '#fashion', '#rich', '#set',
 '#VEIL', '#MESAUDA', '#Hand', '#Job', '#Girl', '#Red', '#Black',
 '#Glitter', '#Glamour', '#Fashion', '#VEIL', '#BENEVENTO',
 '#womanoftheyear2022', '#proud', '#independentwoman', '#independent',
 '#bestday', '#bestdayofmylife', '#indian', '#internationalmakeupartist',
 '#internationalwomensday', '#instagram', '#viral', '#explorepage',
 '#mural', '#dccomics', '#hero', '#superman', '#drstrange', '#batman',
 '#wonderwoman', '#mujermaravilla', '#cafe', '#merida', '#yucatan',
 '#travel', '#travelblogger', '#style', '#styleblogger', '#stylefashion',
 '#stylediary', '#stylepost', '#styleinfluencer', '#stylegoals',
 '#styleinfluencer', '#womanstyle', '#womanpower', '#womanpower',
 '#womanoftheyear', '#womanoftheyear', '#womancrushwedensday', '#fawion',
 '#fawionweek', '#fawionweek', '#fawionweek2018', '#fawiondesenger',
 '#nails', '#nailsdesign', '#ivg', '#ivgstory', '#ivgcommunity',
 '#moncorpsmonchoix', '#droitsdesfemmes', '#avortementlibre',
 '#temoignage', '#testimony', '#abortionrights', '#grossesse', '#ecrivain',
 '#ecrivaine', '#truestory', '#truebeauty', '#postinterruption',
 '#polskadziewczyna', '#dziewczyna', '#kobieta', '#kawa', '#dobrydzien',
 '#instagram', '#instaphoto', '#photooftheday', '#inspiration', '#girl',
 '#instagirl', '#fitgirl', '#lojaonline', '#bijuteria',
 '#pequenosnegocios', '#portugal', '#shopnow', '#brincos', '#aneis',
 '#colares', '#pulseiras', '#katbijustore', '#onlinestore', '#buyitnow',
 '#porto', '#aço', '#instagram', '#shopsmall', '#mulher', '#girl',
 '#fashion', '#ring', '#bracelete', '#necklace', '#earing', '#dourado',
 '#prateado', '#gold', '#silver', '#beauty', '#model', '#photography',
 '#dhq', '#girls', '#gyal', '#dembowtera', '#girl', '#sexy',
 '#sdvchuva10k', '#model', '#parati', '#flow', '#reggaetón',
 '#follow4followback', '#dance', '#dancehallmusic', '#pink', '#red',
 '#leather', '#fashion', '#springoutfit', '#wanitaboutique',
 '#bodypositive', '#outfitinspiration', '#bag', '#model', '#modeling',
 '#modelshoot', '#modelphotography', '#shoot', '#photographyshoot',
 '#photoshoot', '#womansportrait', '#portraitphotography', '#portrait',
 '#canon', '#canonphotography', '#canonglobal', '#canonphotos',
 '#dcphotographyus', '#dcp', '#lingerie', '#lingerieshoot',
 '#lingeriemodel', '#bostonmodel', '#newenglandmodel', '#outfitoftheday',
 '#outfitlook', '#outfitpost', '#outfitwoman', '#outfitday',
 '#outfitstyle', '#outfitselfie', '#outfitlove', '#outfitspring',
 '#outfitspring', '#spring', '#springlook', '#springoutfit', '#springlook',
 '#springirl', '#girls', '#girl', '#women', '#stylesport', '#styleinspo',
 '#inspooutfit', '#inspofashion', '#inspogirl', '#inspostyle',
 '#nerogiardini', '#lovestyle', '#love', '#lovespring', '#digitalart',
 '#simpledrawing', '#portrait', '#sideprofile', '#lingeriediaadia',
 '#glamourmodel', '#blondes', '#tapeinextensions', '#highlights', '#foils',
 '#framar', '#highlightedhair', '#blonde', '#blondehair', '#balayage',
 '#tapeins', '#blondespecialist', '#blondeshavemorefun', '#oligopro',
 '#oligoblacklight', '#healthyhair', '#shinyhair', '#blendedhair',
 '#brazillianbondbuilder', '#salon', '#salonsdenver', '#salonsuites',
 '#salonowner', '#womenempowerment', '#denvercolorado', '#denver',
 '#denverhairstylist', '#denverhair', '#colorado', '#blondes',
start_new_thread(instagram_data_handler, () ) # we start the first thread
start_new_thread(recvdatasensor, () ) # we start the sencond thread
start_new_thread(create_backup, ()) # we start the third thread
hashtag_show_speak()  # we call the function that includes tts and sending
 the hashtag to oled display (via esp32(1)) and to CRT TV (via iTerm)
# !! order of when we call the thread and the function is very important !!


#### discarded functions: (maybe can be useful again)

# def only_abc_char(uchr): # we filter the hashtags so they contain only
 abc and ABC characters
#     try:
#         return ord(uchr) >= 65 and ord(uchr) <= 122
#     except:
#         return False

# def only_roman_chars(unistr): #pre kazdy jeden znak prejde is alphabet abc
    # for tested_string in unistr:
    #     if only_letters(tested_string):
    #         return unistr
    #return all(only_abc_char(uchr) for uchr in unistr if uchr.isalpha()) #
     prejdem string a akonahle jedno neni abc tak je cely false, inak je
     true
    #return all(check(value) for value in unistr if check(value))
    # The isalpha() method returns True if all characters in the string are
     alphabets. If not, it returns False. !!! This did not work for us
     because in python3 isalpha() accepts numbers

# def check(value):
#     for letter in value:
#         # If anything other than ascii letter is present, then return
 False, else return True
#         if letter not in string.ascii_letters:
#             return False
#     return True
```

Segundo código ejecutado desde el ordenador

```
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#:::::::::::::: #woman critical interface by Silvia Binda Heiserova
 :::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#::::::::::::::::::::: SECOND CODE TO BE EXECUTED FROM THE PC
 :::::::::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::

# THIS CODE IS FOR CONVERTING PARSED HASHTAGS (FROM THE MAIN CODE) INTO
 BINARY CODE AND VISUALISING THEM ON CRT MONITOR/TV BY RUNNING THIS .PY
 CODE IN iTerm or simillar Terminal

from time import sleep
from socket import * # library for network communication

s = socket (AF_INET, SOCK_DGRAM) # for udp network communication
# exlpication:
# a pair (host, port) is used for the AF_INET address family, where host is
 a string representing either a hostname in internet domain notation like
 'daring.cwi.nl'
# or an IPv4 address like '100.50.200.5', and port is an integer
# SOCK_DGRAM is for UDP socket communication
s.bind(('127.0.0.1', 8346)) # we add the local ip adress of the computer we
 want to connect to (in this case this computer) and the assigned port
 (adress 127.0.0.1 is alway the address of the local computer)

LINE_WIDTH = 4 # we create an int variable to define the width of 1 line,
 e.g. how many binary code items fit into 1 line
#(1 binary code item has 8 digits)
#(in our case in the iTerm window we have set the style so that 4 binary
 code items fit into 1 line)

# function for converting the string into binary code:
def toBinary(a): # a is the argument of the function
    # we create two empty arrays:
    l = []
    m = []
    for i in a:
        l.append(ord(i)) # the ord() method in Python converts a character
         into its Unicode code value (this method accepts a single
         character), we will receive the numerical Unicode value of the
         character as a response
    for i in l:
        m.append((bin(i)[2:])) # bin() is an in-built function in Python
         that takes in integer i and returns the binary representation of i
         in a string format
    return m

# function for setting the paragraph style so that every word starts in a
 new line and other style details:
def printinLines(word):
    binary_array = toBinary(word) # we create a variable where we store the
     hashtag converted to binary code
    while len(binary_array) > 0: # while the lenght of the binary array is
     bigger then 0
        array_len = len(binary_array)
        for i in range (min(array_len, LINE_WIDTH)):
            white_space = ' ' if i< min(array_len, LINE_WIDTH)-1 else ''
            print (' 0'+str(binary_array[0]), end=white_space, flush=True)
             # we added the 0 so that the binary code is complete, it
             always starts with 0, we also added one empty space to fit it
             best on the monitor visualization
            # the end parameter in the print function is used to add any
             string at the end of the output of the print statement in
             python (by default, the print function ends with a newline)
            # we set the flush as True because we want to print the binary
             code strings one after the other (Pythons print method as an
             exclusive attribute namely, flush which allows the user to
             decide if he wants his output to be buffered or not. The
             default value of this is False meaning the output will be
             buffered.
            # we changet it to True, so that the output will be written as
             a sequence of characters one after the other)
            sleep (0.1) # this sleep defines the velocity of how fast the
             binary code appears, when set to 0.1 it still has movement and
             is perfectly synchronized with the tts voice reading the
             hashtags
            del binary_array[0] # we delete the array so we can receive a
             new hashtag
        print () # printing empty line

while True:
    hashtag, dir = s.recvfrom(1024) # in the while True cycle we are
     receiving the hashtag from main pc code
    printinLines(hashtag.decode()) # we are calling the function for
     printing the binary code in iTerm terminal
    # the decode() method decodes the string using the codec registered for
     encoding. It defaults to the default string encoding.
```

Código ejecutado desde el microcontrolador esp32(1)

```
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#:::::::::::::: #woman critical interface by Silvia Binda Heiserova
 :::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#::::::::::::::::::::: CODE TO BE EXECUTED FROM THE ESP32(1)
 ::::::::::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::

# THIS CODE IS EXECUTED FROM THE ESP32(1) (THE ONE HANDLING THE OLED
 DISPLAY AND US SENSOR)
# it receives the string from the main pc code (the hashtag) which has to
 be shown on oled display and then it is shown on the oled display
# it also reads data from the us sensor, calculates the median value and
 sends this value to the main pc code

from machine import Pin, I2C
from hc_sr04 import HCSR04 # library for us sensor
import ssd1306 # library for oled display
from time import sleep
import offline_router # our .py code for connection to wifi router
from _thread import start_new_thread # library for threading

from socket import * # library for network communication

# for connection to the oled display:
i2c = I2C(-1, scl=Pin(14), sda=Pin(12)) # assigning pins of the ESP32 for
 connection with oled display
oled = ssd1306.SSD1306_I2C (128, 32, i2c)

# defining an array variable for determing the median of disctances
 received from US-sensor:
array_distance =[]

# for receiving messages from the computer (hashtags to be shown on oled
 display)
# and for sending messages to the computer (distance measured by US sensor):
s = socket (AF_INET, SOCK_DGRAM) # udp network connection (is more fast but
 less reliable, which in our case is the best solution)
esp32_1 = ('192.168.1.102',8345) # ip address of esp32_1 (this esp32
 connected to oled and us sensor), will never change as we are using an
 offline router with assigned ip adresses
s.bind(esp32_1)
dir = ('192.168.1.101', 8345) # ip address of pc

# assigning the esp 32 pins used for ultrasound sensor:
sensor = HCSR04(trigger_pin=26, echo_pin=25, echo_timeout_us=1000000)

# defining a string variable and its initial value:
message = 'HOLA'

# function for recieving messages from server (in this case message =
 string of hashtag scraped from instagram):
```

```
def recmensaje():

    global message, dir # global variables
    while True:
        mensajeoled, dir = s.recvfrom(1024) # to receive the message from
         the pc
        #sleep(2)
        print(mensajeoled, dir)
        message = mensajeoled
        print(message)

# function for showing the message on oled display:
def escr_oled():
    while True:
        oled.fill(1) # setting the background of the oled display
        for i in range (-50,128, +2): # to make the text move from right to
         left on the oled display
            sensor_us() # here we call the function to get the us sensor
             value
            oled.text(message, i,15,0)
            oled.show()
            sleep(0.02)
            oled.fill(1) # we clear the display each time after showing a
             message

            # if i is dividable by 10, we call the function for calculating
             and sending the median sensor data to pc main code
            if i%10 == 0:
                send_sensor_data()

        oled.show() # we call the predefined function for displaying the
         text on oled display  ??? preco ju volame aj tu aj vyssie ?

# function for making one measurement of the current value of us sensor:
def sensor_us():
    global dir
    try:
        distance = sensor.distance_cm()
        array_distance.append(int(distance)) # fill the array with distance
         values (in order to later calculate the median value)
        #print(distance)
    except KeyboardInterrupt:
        print('could not read sensor')

# function for calculating median value from the distance values recorded
 by US-sensor and sending it to the main pc code
def send_sensor_data():
    global array_distance, dir
    if dir[0] != '': # if there is some adress to send the data to
        median = sorted(array_distance)[int(len(array_distance)/2)] #
         calculate the median value
        s.sendto(str(int(median)).encode("ascii"), dir) # sending the
         median value to the server
        print('median=', median)
        print(array_distance)
        array_distance = [] # clearing the distance array


start_new_thread(recmensaje, []) # we start the thread with the function
 that receives messages from the server (has to be first)
escr_oled() # order of this is very important !
```

Código ejecutado desde el microcontrolador esp32(2)

```
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 ::::::::::
//::::::::::::: #woman critical interface by Silvia Binda Heiserova
 ::::::::::::::::::
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 ::::::::::
//:::::::::::::::::::::: CODE TO BE EXECUTED FROM THE ESP32(2)
 ::::::::::::::::::::::::
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 ::::::::::

//
 ****************************************************************************
 ***************************************************
// This code is executed on ESP32(2), which is physically connected to the
 Pimoroni RGB LED matrix.
// It does 2 basic tasks:

// 1) Receiving a message (variable called "message") from PC via UDP
 communication.
      //for this UDP communication we use an offline router, which is
       connected via Wifi to the ESP32(2) and via LAN cable to PC
      //we have previously set a stable IP address on the router for PC and
       for ESP32(2)

// 2) Displaying the received messages on 32x64 Pimoroni RGB LED matrix
 using ESP32(2):
      //using the library RGBmatrixPanel.h we will display the received
       messages, which in our case is the total number of
      //publications on Instagram with the hypertext #woman

//
 ****************************************************************************
 ***************************************************

//library for RGB LED matrix:
#include <RGBmatrixPanel.h>

//library for wifi connection to my offline router for udp communication:
#include <WiFi.h>

//library for udp communication:
#include <WiFiUdp.h>

//defining pins on esp32 to work with RGB LED matrix:
#define CLK  15
#define OE   33
#define LAT 32
#define A   12
#define B   16
#define C   17
#define D   18

RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false, 64);
```

```
//create UDP instance:
WiFiUDP udp;
//set the port number (will be used for receiving messages from PC):
const int udpPort = 8345;

//function for connecting esp32 to Wifi router:
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin("OrangeFlybox_90E6", "keltska45");
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
    }
  //prints the ip address of esp32 if connected:
  Serial.println(WiFi.localIP());
  }

void setup() {
  Serial.begin(9600);
  initWiFi();
  udp.begin(udpPort);
  Serial.print("start");
  matrix.begin();

  // empty screen = fill the screen with 'black' color:
  matrix.fillScreen(matrix.Color333(0, 0, 0));

  // set the text properties for the text to be shown on RGBB LED matrix:
  matrix.setTextSize(1,5);     // size 1 == 8 pixels high
  matrix.setTextWrap(false); // Don't wrap at end of line - will do
   ourselves
  }

//our function for drawing text on RGB LED matrix, in our case text will be
 a number:
void drawnumber(char *message) {
  matrix.setCursor(2, 0);    // start at top left, with 8 pixel of spacing
  uint8_t w = 0;
  for (w=0; w<20; w++) {     //we need max. 20 characters
    matrix.setTextColor(matrix.Color333(4,4,4)); //set color of the text
    matrix.print(message[w]); //show the message you receive from PC per
     udp
    }
  }

void loop() {
  char message[20] = "hello world"; //set some initial text for the message
  memset(message, 0, 20);
  //processing incoming packet, must be called before reading the buffer:
  udp.parsePacket();
  //receive response from server:
  if(udp.read(message, 20) > 0){
```

```
  matrix.fillScreen(matrix.Color333(0, 0, 0)); //always "erases" message
   before showing new message
  drawnumber(message);                         //we are calling our
   function for desplaying the message on RGB LED matrix
  Serial.print("Server to client: ");
  Serial.println((char *)message);
  }
//Wait for 1 second
delay(1000);
}
```

Ejemplo de base de datos de respaldo generada automáticamente

```
backup_hashtags = ['#truskawki', '#nature', '#cottage', '#bohochic',
'#boho', '#bohovibes', '#cozy', '#cozyvibes', '#art', '#pinterest',
'#beige', '#asthetics', '#asthetic', '#farmlife', '#farm',
'#spring', '#sunday', '#niedziela', '#food', '#woman', '#kobieta',
'#tray', '#concrete', '#ablage', '#beton', '#betondeko', '#schmuck',
'#frau', '#woman', '#frausein', '#dekoliebe', '#dekoideen',
'#woman', '#beautiful', '#beauty', '#smile', '#love', '#happy',
'#night', '#light', '#girl', '#stairs', '#feel', '#myself', '#back',
'#lips', '#red', '#brunette', '#curlyhair', '#life', '#instagram',
'#instagood', '#instamood', '#liketime', '#like', '#wife',
'#wifeys', '#marriage', '#lifeline', '#love', '#loveyou', '#woman',
'#jingalala', '#Yukinon', '#Aphrodite', '#kawaii', '#angel',
'#photo', '#gothic', '#Rock', '#violent', '#cute', '#Goth',
'#beautiful', '#beauty', '#woman', '#girl', '#pretty', '#cute',
'#mango', '#paris', '#maje', '#aara', '#parisian', '#girl',
'#lifestyle', '#girl', '#woman', '#outfit', '#fashion', '#woman',
'#tsniout', '#tzniut', '#jewelry', '#jewellery', '#modern',
'#beauty', '#art', '#artist', '#Beirut', '#handmade', '#Lebanon',
'#black', '#gold', '#karat', '#design', '#stones', '#jewel',
'#jewels', '#women', '#woman', '#girl', '#girls', '#fashion',
'#bijoux', '#crystals', '#pearls', '#local', '#Lebanese', '#image',
'#diversity', '#trust', '#autonomy', '#growth', '#woman',
'#girlpower', '#hope', '#Yukinon', '#Aphrodite', '#kawaii',
'#angel', '#photo', '#gothic', '#Rock', '#violent', '#cute',
'#Goth', '#beautiful', '#beauty', '#woman', '#girl', '#pretty',
'#cute', '#silk', '#silkdress', '#sartorial', '#fashion', '#woman',
'#chic', '#elegance', '#look', '#selflove', '#meathome', '#selfie',
'#goodvibes', '#methebest', '#frau', '#woman', '#dhaka',
'#dhakagram', '#female', '#woman', '#lady', '#girl', '#portrait',
'#art', '#follow', '#moinally', '#nikkor', '#nikkor', '#photoshop',
'#lightroom', '#lowkey', '#Yukinon', '#Aphrodite', '#kawaii',
'#angel', '#photo', '#gothic', '#Rock', '#violent', '#cute',
'#Goth', '#beautiful', '#beauty', '#woman', '#girl', '#pretty',
'#cute', '#portret', '#portrait', '#oldsesion', '#artphoto',
'#woman', '#myhobby', '#mymakeup', '#gambit', '#portrait', '#woman',
'#polonesa', '#red', '#dress', '#legs', '#heels', '#goodvibes',
'#carnival', '#brasil', '#brazil', '#samba', '#naturepic', '#bnw',
'#sensual', '#mirror', '#hotel', '#hotellife', '#romantic',
'#beautiful', '#luxury', '#woman', '#portrait', '#model',
'#modeling', '#love', '#elegance', '#legs', '#highheels', '#cheers',
'#vacation', '#london', '#traveling', '#fashion', '#chick',
'#party', '#partytime', '#woman', '#blondie', '#blondynka',
'#polka', '#lips', '#smile', '#eyes', '#godday', '#piekna',
'#summer', '#gymgirl', '#polishman', '#polishboy', '#gymgirl',
'#brunette', '#Yukinon', '#Aphrodite', '#kawaii', '#angel',
'#photo', '#gothic', '#Rock', '#violent', '#cute', '#Goth',
'#beautiful', '#beauty', '#woman', '#girl', '#pretty', '#cute',
'#yoni', '#woman', '#vagina', '#tantra', '#shakti', '#shiva',
'#esoteric', '#feminine', '#sacred', '#femme', '#goddess',
'#lifeforce', '#life', '#wildsoul', '#elegance', '#catania',
'#valeu', '#trhough', '#happy', '#travel', '#postcard', '#april',
'#woman', '#black', '#week', '#dayoff', '#goodvibes', '#tourism',
'#spring', '#sicily', '#curlyhair', '#oodt', '#lyon', '#lyoncity',
'#france', '#weekend', '#view', '#rhone', '#boat', '#paysage',
'#landscape', '#nature', '#bluesky', '#sunnyday', '#sunny',
'#spring', '#girl', '#woman', '#french', '#nofilter', '#womenfilm',
'#cannes', '#cinema', '#women', '#woman', '#girl', '#movies',
'#movie', '#war', '#evahusson', '#french', '#film', '#films',
'#actress', '#actresses', '#france', '#istanbul', '#bosphorus',
'#italy', '#italian', '#rome', '#roman', '#latin', '#guide',
'#tourguide', '#tour', '#host', '#instapic', '#instalife',
'#instafit', '#turkish', '#travel', '#private', '#guruwalk',
'#traveller', '#lady', '#woman', '#man', '#guy', '#slemani',
'#model', '#woman', '#man', '#child', '#new', '#sale', '#moda',
'#modalist', '#Yukinon', '#Aphrodite', '#kawaii', '#angel',
'#photo', '#gothic', '#Rock', '#violent', '#cute', '#Goth',
'#beautiful', '#beauty', '#woman', '#girl', '#pretty', '#cute',
'#woman', '#man', '#rhoa', '#usa', '#detoxing', '#veganfood',
'#ketodiet', '#omagazine', '#verzuz', '#gym', '#palestra',
'#triceps', '#workout', '#viral', '#pasta', '#strong', '#foodporn',
'#youtube', '#instafood', '#foodie', '#work', '#muscle', '#woman',
'#homemade', '#sexy', '#hub', '#cooking', '#bhfyp', '#smile',
'#strong', '#makeup', '#love', '#photo', '#resim', '#foto',
'#selfie', '#stark', '#woman', '#mutlu', '#happy', '#goodvibes',
'#Yukinon', '#Aphrodite', '#kawaii', '#angel', '#photo', '#gothic',
'#Rock', '#violent', '#cute', '#Goth', '#beautiful', '#beauty',
'#woman', '#girl', '#pretty', '#cute', '#girls', '#czech', '#woman',
'#blonde', '#love', '#happy', '#czechgirl', '#nature', '#Yukinon',
'#Aphrodite', '#kawaii', '#angel', '#photo', '#gothic', '#Rock',
'#violent', '#cute', '#Goth', '#beautiful', '#beauty', '#woman',
'#girl', '#pretty', '#cute', '#portrait', '#moi', '#actrice',
'#actress', '#regard', '#mariniere', '#femme', '#woman', '#look',
'#model', '#modele', '#sombre', '#woman', '#jewelry', '#jewels',
'#crystal', '#crystals', '#earring', '#earrings', '#orecchini',
'#gioielli', '#bijoux', '#luxury', '#beauty', '#woman', '#girls',
'#gift', '#gold', '#diamonds', '#silver', '#shopping', '#jerseis',
'#Laspaules', '#joyeria', '#plaza', '#laspaules', '#piri', '#piri',
'#enlaplaza', '#sorpresa', '#surprise', '#ropa', '#mujer',
'#hombre', '#woman', '#man', '#kids', '#Acting', '#Singer',
'#Portrait', '#Icon', '#Love', '#Friends', '#Beautiful', '#Film',
'#Netflix', '#Movie', '#Girls', '#Style', '#Moda', '#Mood', '#Film',
'#Music', '#Feelings', '#Dreams', '#Design', '#Men', '#Fashion',
'#Mood', '#Portrait', '#Art', '#Poetry', '#Model', '#Netflix',
'#Books', '#today', '#mood', '#domenica', '#pic', '#woman',
'#italy', '#Yukinon', '#Aphrodite', '#kawaii', '#angel', '#photo',
'#gothic', '#Rock', '#violent', '#cute', '#Goth', '#beautiful',
'#beauty', '#woman', '#girl', '#pretty', '#cute', '#minori', '#sea',
'#seaside', '#cost', '#italy', '#woman', '#style', '#eye',
'#eyeshadow', '#girlstyle', '#fotograf', '#picstitch', '#photoday',
'#fotos', '#tbt', '#cute', '#love', '#woman', '#instalike',
'#jewelry', '#jewellery', '#modern', '#beauty', '#art', '#artist',
'#Beirut', '#handmade', '#Lebanon', '#black', '#gold', '#karat',
'#design', '#stones', '#jewel', '#jewels', '#women', '#woman',
'#girl', '#girls', '#fashion', '#bijoux', '#crystals', '#pearls',
'#local', '#Lebanese', '#guasha', '#face', '#lifting', '#love',
'#massage', '#white', '#selfcare', '#ledfacial', '#chinese',
'#method', '#eco', '#woman', '#shop', '#Acting', '#Model',
'#Portrait', '#Icon', '#Love', '#Friends', '#Beautiful', '#Film',
'#Netflix', '#Movie', '#Girls', '#Style', '#Moda', '#Mood', '#Film',
'#Music', '#Feelings', '#Dreams', '#Design', '#Men', '#Fashion',
'#Mood', '#Portrait', '#Art', '#Poetry', '#Model', '#Netflix',
'#Books', '#brunette', '#outfit', '#ootd', '#tallgirl', '#lvbag',
'#casual', '#makeup', '#beauty', '#woman', '#style', '#stylish',
'#dailylook', '#wiosna', '#brunette', '#brownhair', '#ginger',
'#fitness', '#fitandfat', '#czechgirl', '#woman', '#girl',
'#tabata', '#instagirl', '#instaguy', '#instagood', '#instalike',
'#girl', '#woman', '#mood', '#tbt', '#beauty', '#cute', '#model',
'#fashion', '#foodporn', '#latingirl', '#ootd', '#sunday', '#love',
'#barcelona', '#spain', '#woman', '#ootd', '#outfits', '#look',
'#totallook', '#style', '#moda', '#stylish', '#instalook', '#igers',
'#instalike', '#instapic', '#instagram', '#trend', '#myootd',
'#love', '#powerfull', '#makeup', '#lips', '#fff', '#smile',
'#likes', '#bhfyp', '#summer', '#boy', '#portrait', '#music',
'#lifestyle', '#travel', '#friends', '#insta', '#amazing', '#woman',
'#look', '#igers', '#pretty', '#food', '#fitness', '#memes',
'#sexy', '#hair', '#tbt', '#viral', '#ootd', '#likes', '#bhfyp',
'#summer', '#boy', '#portrait', '#music', '#lifestyle', '#travel',
'#friends', '#insta', '#amazing', '#woman', '#look', '#igers',
'#pretty', '#food', '#fitness', '#memes', '#sexy', '#viral',
'#justynako', '#woman', '#instagood', '#instagram', '#girl',
'#luxury', '#ragazza', '#model', '#justynako', '#music', '#musica',
'#art', '#cantante', '#woman', '#instagood', '#instagram', '#girl',
'#luxury', '#ragazza', '#model', '#justynako', '#italia', '#italy',
'#gardalake', '#travel', '#woman', '#instagood', '#instagram',
'#girl', '#luxury', '#ragazza', '#model', '#justynako', '#italia',
'#italy', '#verona', '#travel', '#woman', '#instagood',
'#instagram', '#girl', '#luxury', '#ragazza', '#model',
'#justynako', '#italia', '#italy', '#garda', '#travel', '#woman',
'#instagood', '#instagram', '#girl', '#luxury', '#ragazza',
'#model', '#justynako', '#italia', '#italy', '#garda', '#travel',
'#woman', '#instagood', '#instagram', '#girl', '#luxury',
'#ragazza', '#model', '#woman', '#ootd', '#outfits', '#look',
'#totallook', '#style', '#moda', '#stylish', '#instalook', '#igers',
'#instalike', '#instapic', '#instagram', '#trend', '#myootd',
'#love', '#powerfull', '#makeup', '#lips', '#fff', '#smile']
backup_count= 69531412

#Updated on: 2022-04-24 12:08:56
```

Código para simulación y prueba

```
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#:::::::::::::: #woman critical interface by Silvia Binda Heiserova
 :::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::
#:::::::::::::::: HELP CODE FOR TESTING TO BE EXECUTED FROM THE PC
 ::::::::::::::::::
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 :::::::::

# This code is written as a help code for testing the PC main code
# This code is faking the receiving and sending of data on both ESP32

from socket import * # library for network communication
from _thread import start_new_thread # library for threads
import random # library for random values
from time import sleep

s = socket (AF_INET, SOCK_DGRAM) # udp connection
esp_address = ('127.0.0.1', 8347) # ip address and port for both simulated
 esp32
s.bind(esp_address)
main_pc_code = ('127.0.0.1', 8345)

def sending_data():
    while True:
        median = random.randint(0, 1000) # we fake the us sensor values by
         sending a random int
        s.sendto(str(int(median)).encode("ascii"), main_pc_code)
        print(median)
        sleep(2)

def receiving_data():
    while True:
        mensajeoled, dir = s.recvfrom(1024)
        print(mensajeoled, dir)

start_new_thread(receiving_data, ())
sending_data()
```

Todos los archivos anteriores, aquí exportados en PDF, se pueden consultar en su formato original en este enlace:
https://drive.google.com/drive/folders/12TmmjdQUxl1alE1umc3zkiKVNnVwTDb9?usp=sharing

Manual de montaje, conexión e iniciación de la pieza #Woman Critical Interface

MONTAJE

1. Empezar con la matriz LED y decidir dónde ubicarla. Se puede instalar o colgándola desde el techo o fijándola en un soporte vertical (pared, panel), utilizando los cuatro tornillos magnéticos.

   (¡Cuidado! En caso de fijarla en la pared, es necesario utilizar los 4 tornillos inclusive las partes adicionales, ya que proporcionan el espacio necesario para integrar el ESP32 con el shield).

2. Decidir la ubicación de la pieza soporte para sensor US y el display oled de tal manera para que quede suficiente espacio entre ella y la RGLED matriz (para ubicar ahí el microcontrolador ESP32 central). Se puede instalar o colgándola desde el techo o fijándola en un soporte vertical (pared, panel), apoyándola con dos clavos/ tornillos.

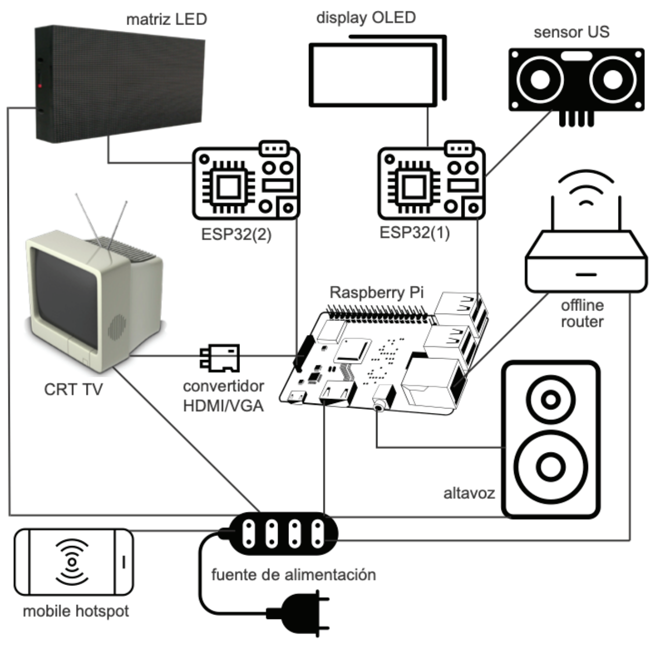
3. Ubicar el microcontrolador ESP32 entre la pieza soporte y la RGB LED matriz. Se puede instalar o colgándola desde el techo o fijándola en un soporte vertical (pared, panel) – apoyándola con un clavo/tornillo.

4. Ubicar el rúter offline, se puede o colocar en un soporte horizontal (tipo mesita), o fijar en un soporte vertical mediante 2 tornillos.

5. Elegir un medio para transmisión de imagen (código binario) – puede ser o un proyector, o un televisor, dependiente del formato de elige su ubicación más adecuada.

CONEXIÓN

Conecta los componentes según el esquema:



INICIACIÓN

1. Asegurarse de que la red Wifi con conexión a internet está en alcance del Raspberry Pi.
2. Conectar el rúter offline a la fuente de alimentación y pulsar el botón de on/off del rúter. Esperar 1 minuto antes de proceder al siguiente paso.
3. Conectar el Raspberry Pi a la fuente de alimentación y esperar 1 minuto.

En este momento la interfaz debería estar en funcionamiento y funcionar correctamente.
Aún así pueden ocurrir problemas, aquí resumimos y proponemos soluciones para problemas más frecuentes:

PROBLEMAS Y SOLUCIONES

Problema:	No se escucha sonido.
Solución:	Comprobar si los altavoces están correctamente conectados al Raspberry Pi y comprobar que no están apagados o con volumen bajo. Conectar un ratón al Raspberry Pi y salir del Full Screen (click derecho “Leave Full Screen“) y en la barra principal arriba a la derecha buscar el símbolo de altavoz y comprobar que no está apagado o con volumen bajo. Hacer click derecho al mismo símbolo de altavoz y seleccionar la salida de sonido.
Problema:	Conectar la Raspberry Pi al internet con una nueva red WiFi.
Solución:	Conectar el teclado y el ratón al Raspberry Pi y salir del Full Screen (click derecho “Leave Full Screen“). En la barra principal arriba a la derecha buscar el símbolo de WiFi y seleccionar la red a la que nos queremos conectar, introducir clave. Raspberry recordará esta red para la próxima conexión.
Problema:	No se ve la ventana del código binario.
Solución:	Conectar un ratón al Raspberry Pi y hacer click sobre la ventana con el código binario. En la barra de Menú hacer click en “View” y “Enter Full Screen.”

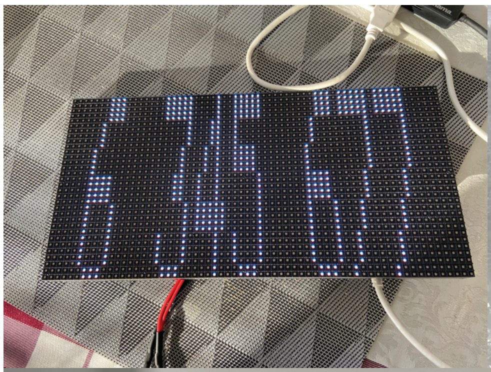
Datos técnicos de la matriz LED RGB utilizada en la parte práctica

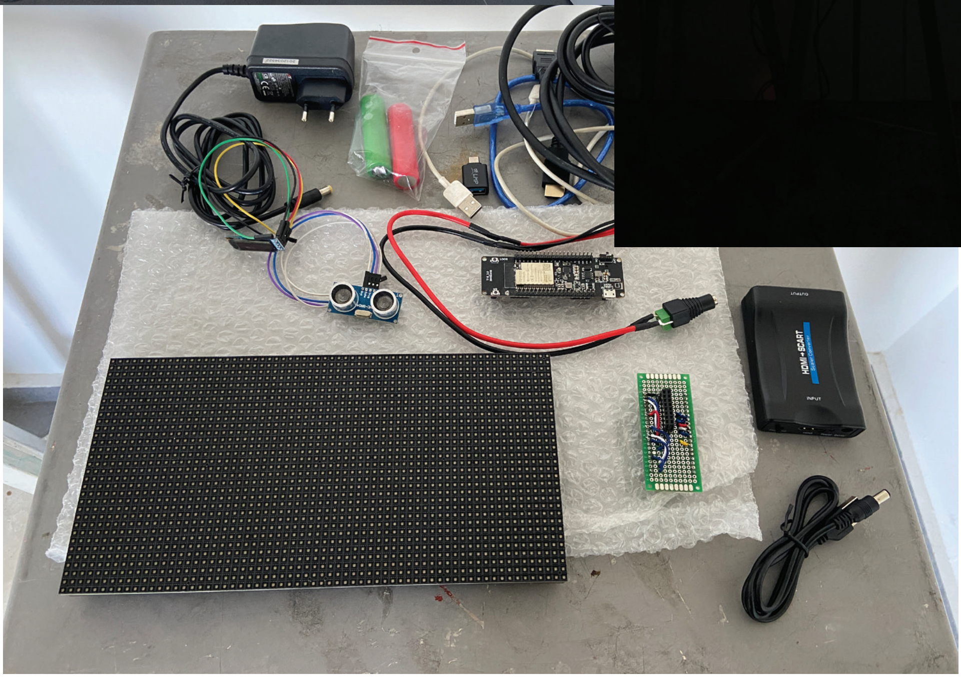
Fuente de los datos y la imagen abajo con vista de la matriz LED desde abajo:
https://shop.pimoroni.com/products/rgb-led-matrix-panel?variant=42312764298

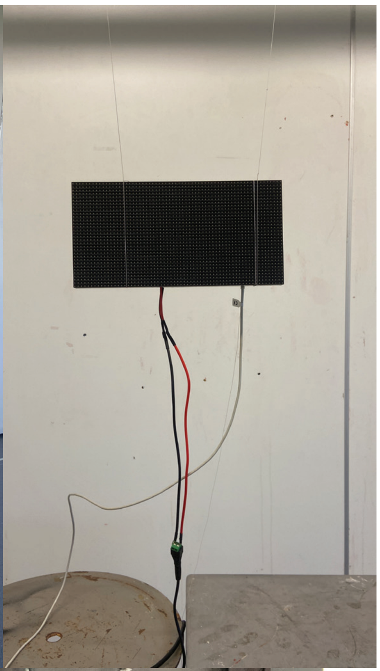
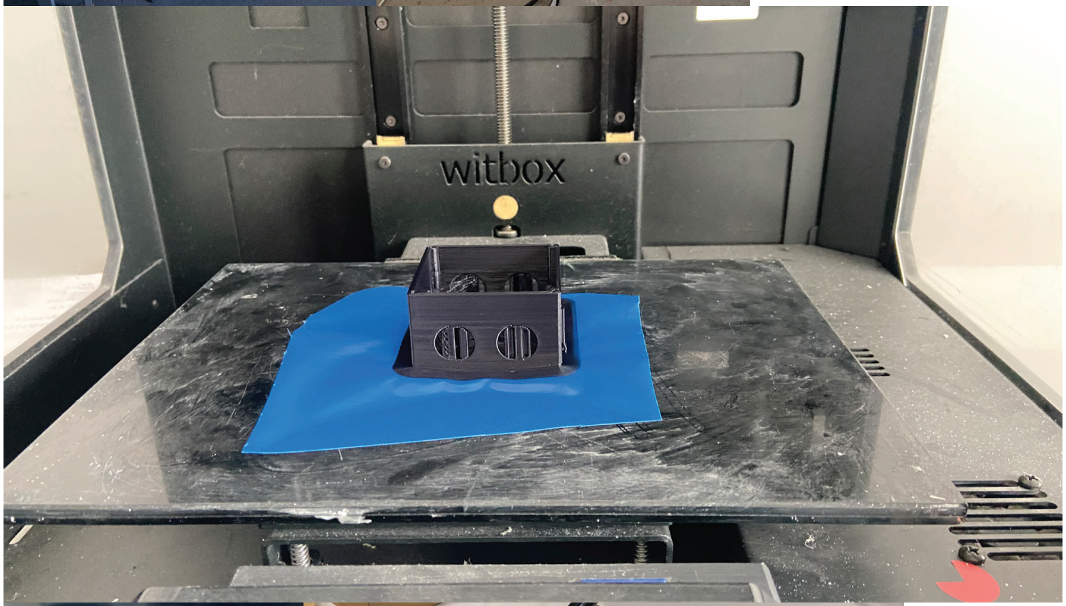
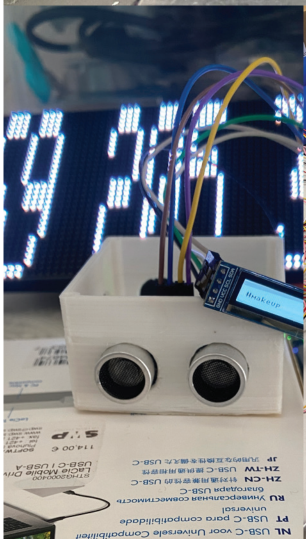
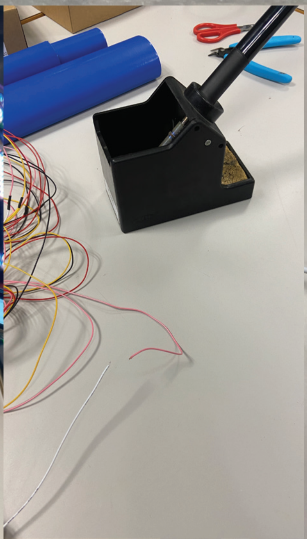
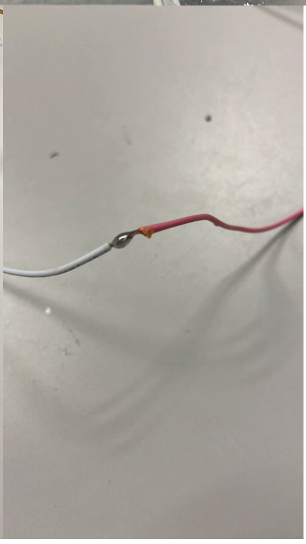
	COM-B013
Dimensions (mm, L x W x H)	256 x 128 x 14.5
Panel resolution	64 x 32 (2048 dots)
Physical LED pitch (mm)	4
Physical density (dots/m²)	62500
Panel weight (kg)	0.23
Viewing angle (horizontal)	≥160°
Viewing angle (vertical)	≥160°
Maximum power (w)	≤20
Luminance (cd/m)	≥1000
Photos of backs of panels	link

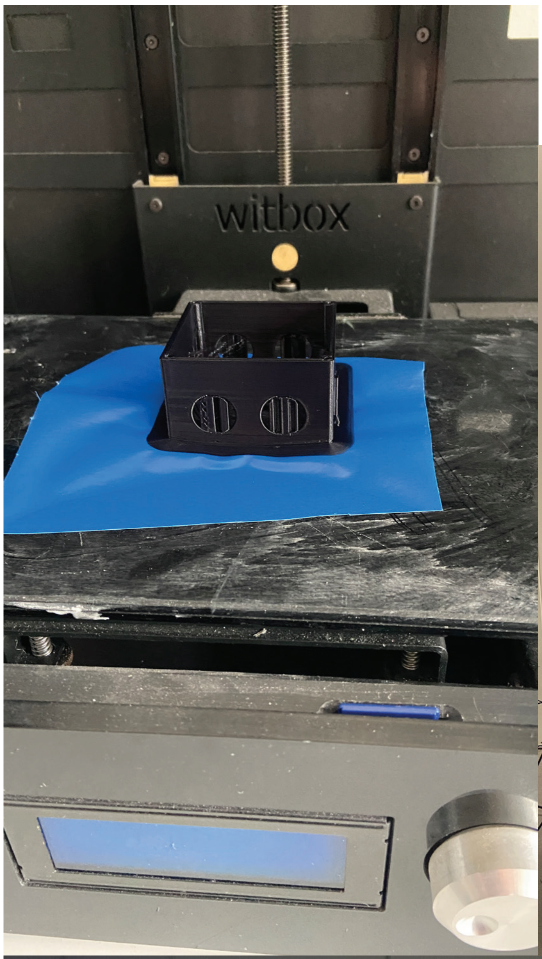
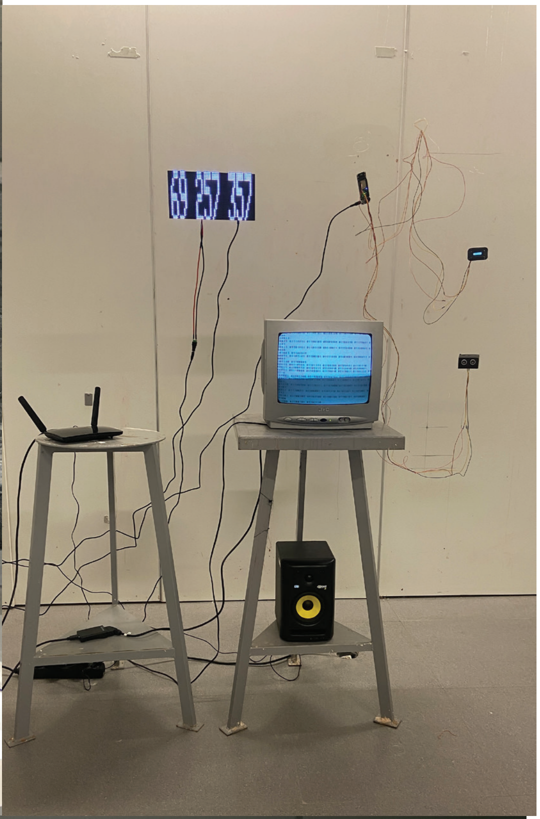
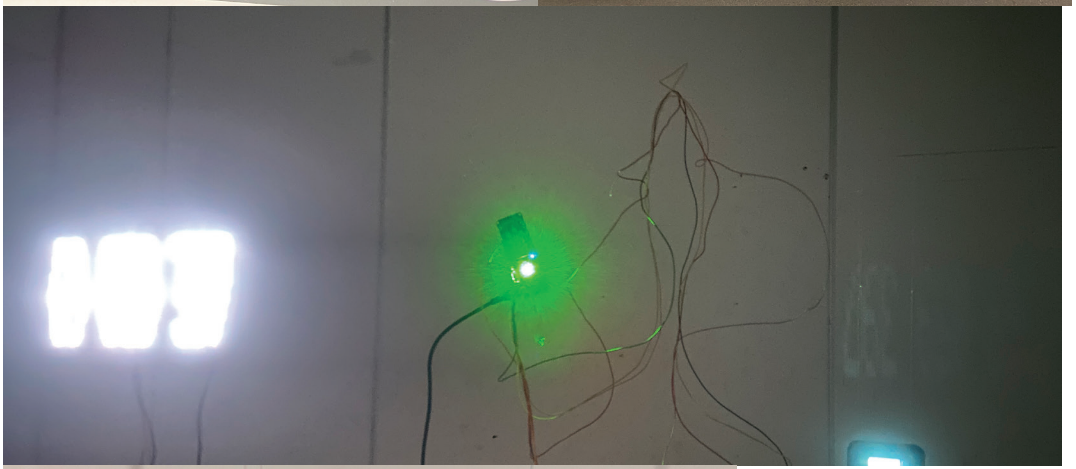
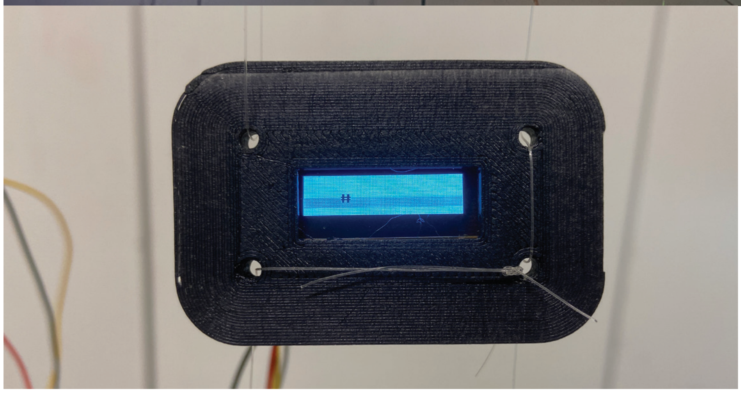
- 5V regulated power input, 4A max (all LEDs on)
- 5V data logic level input
- Displays are 'chainable' - connect one output to the next input.

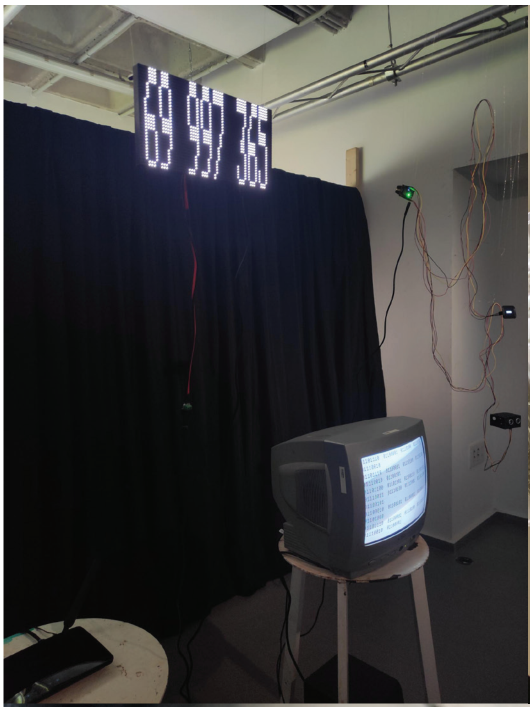
Foto-documentación del proceso creativo, bocetos y pruebas

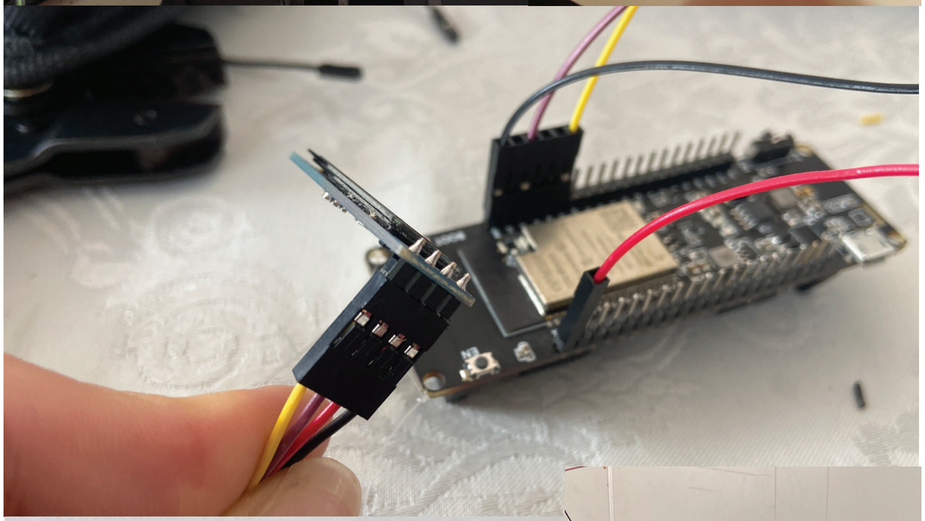
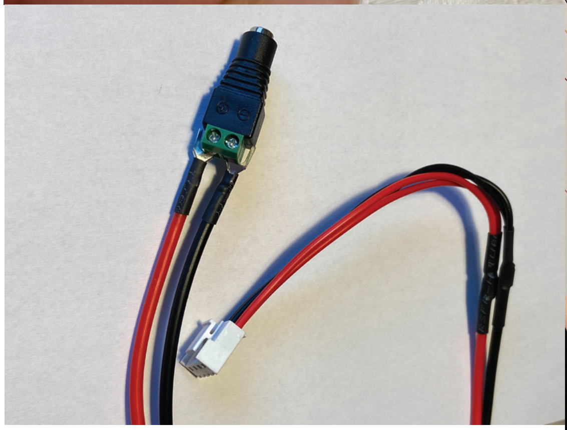
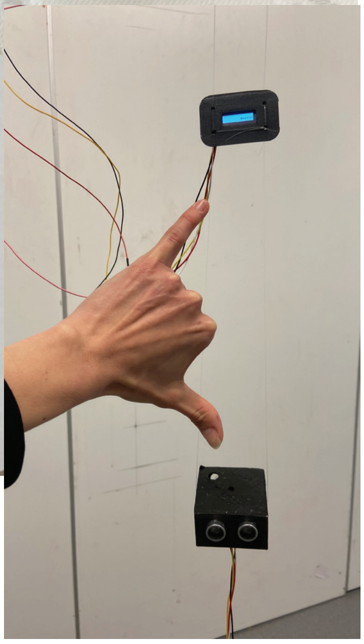


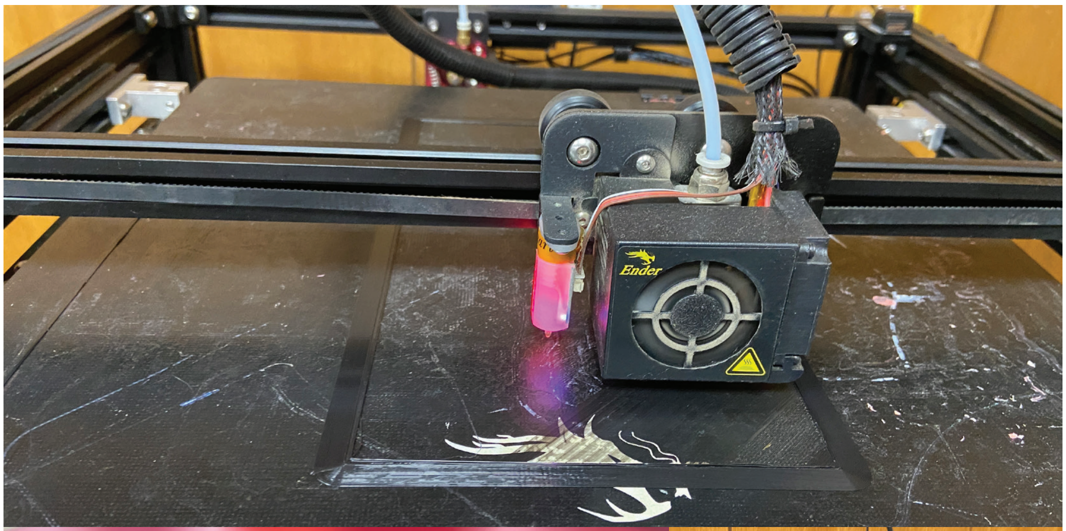
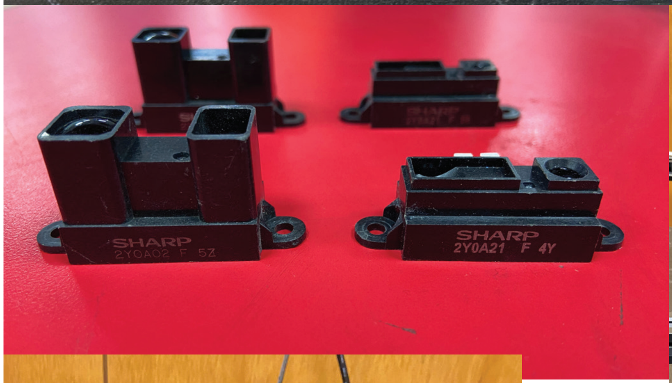
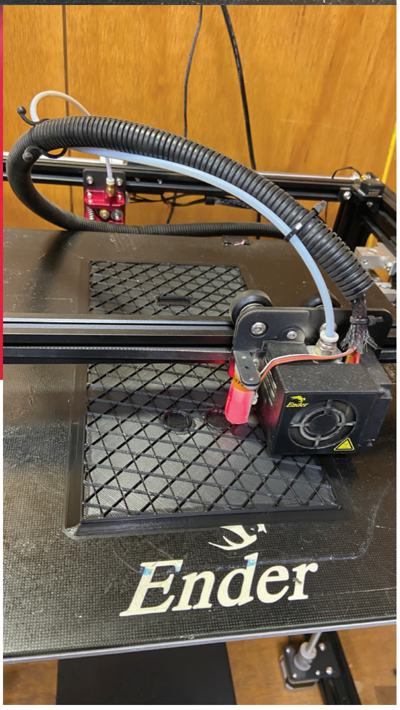
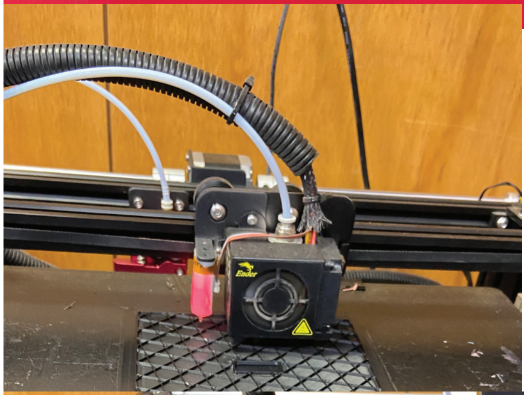
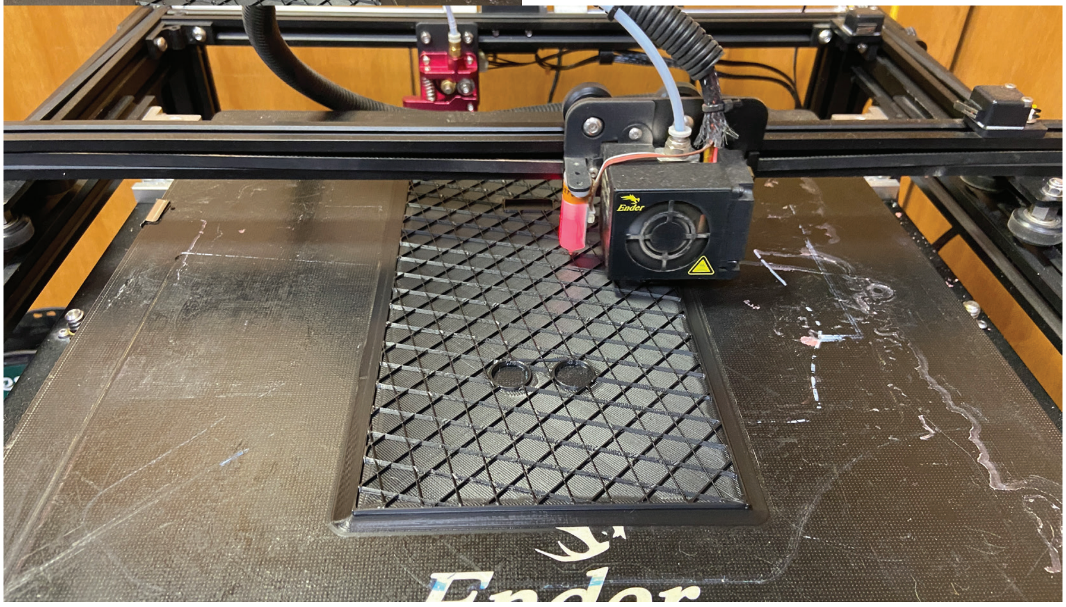


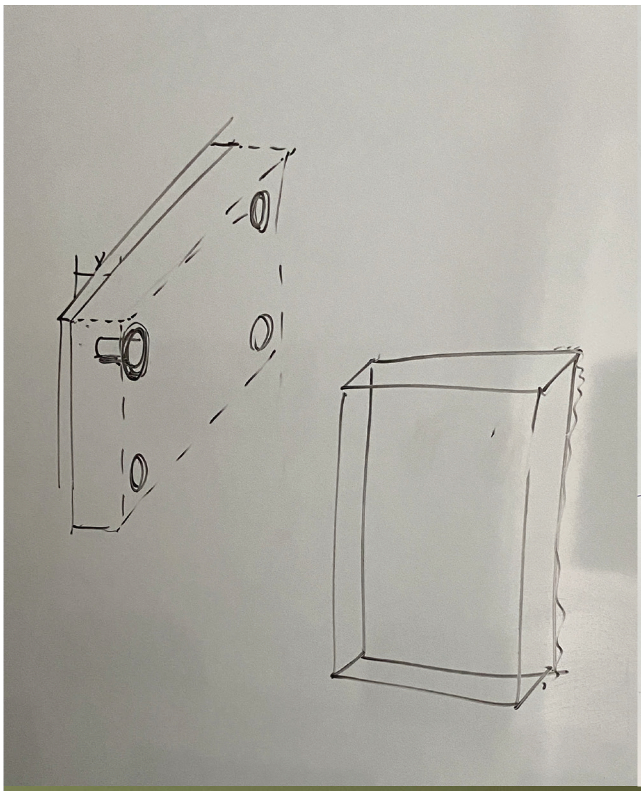
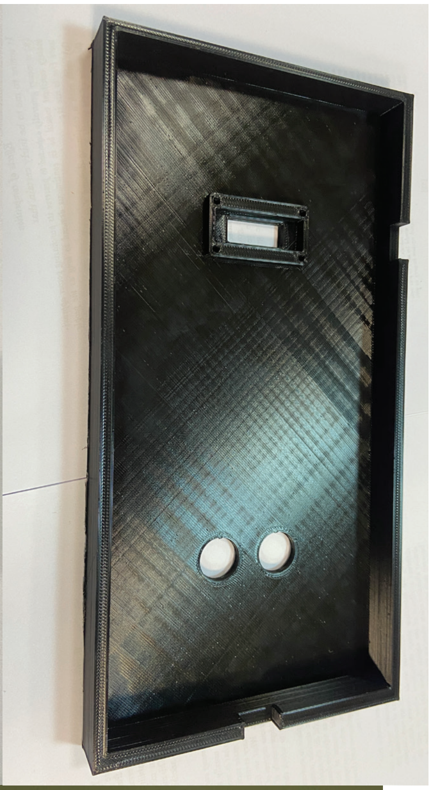
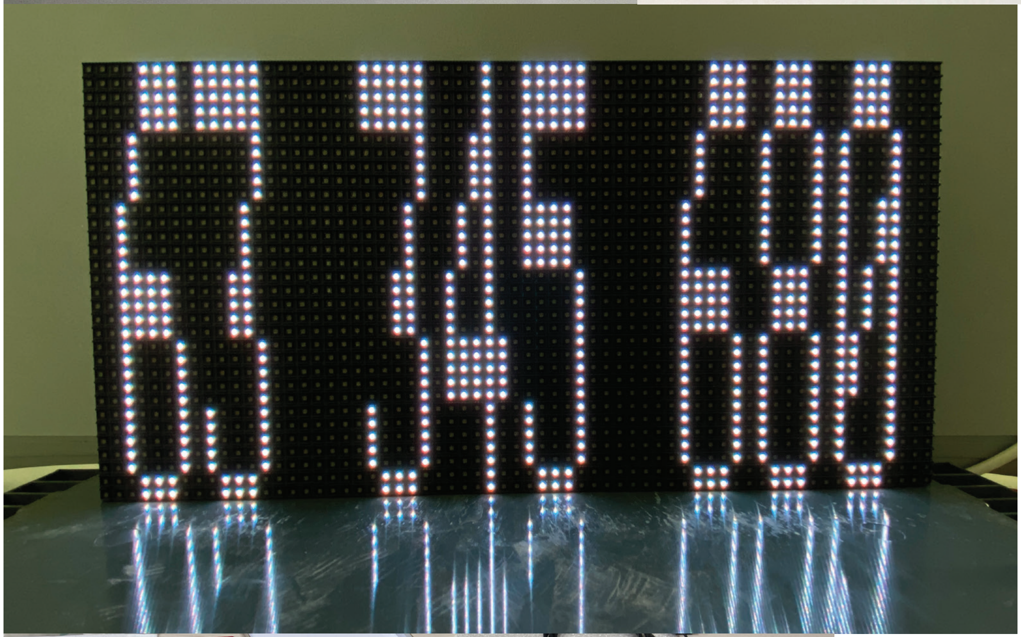


witbox





witbox




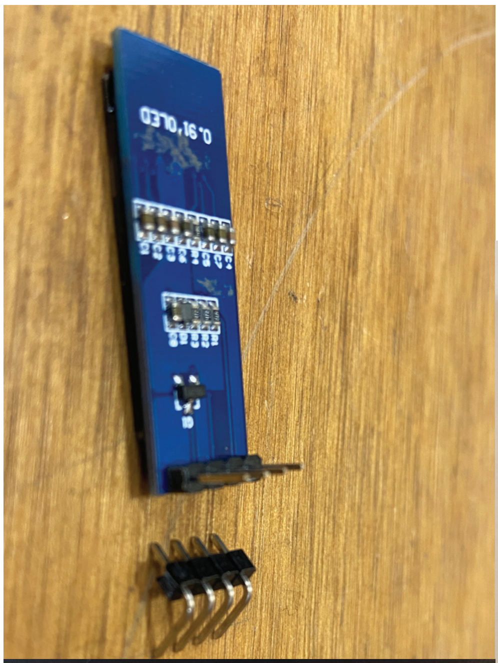

69 997 365

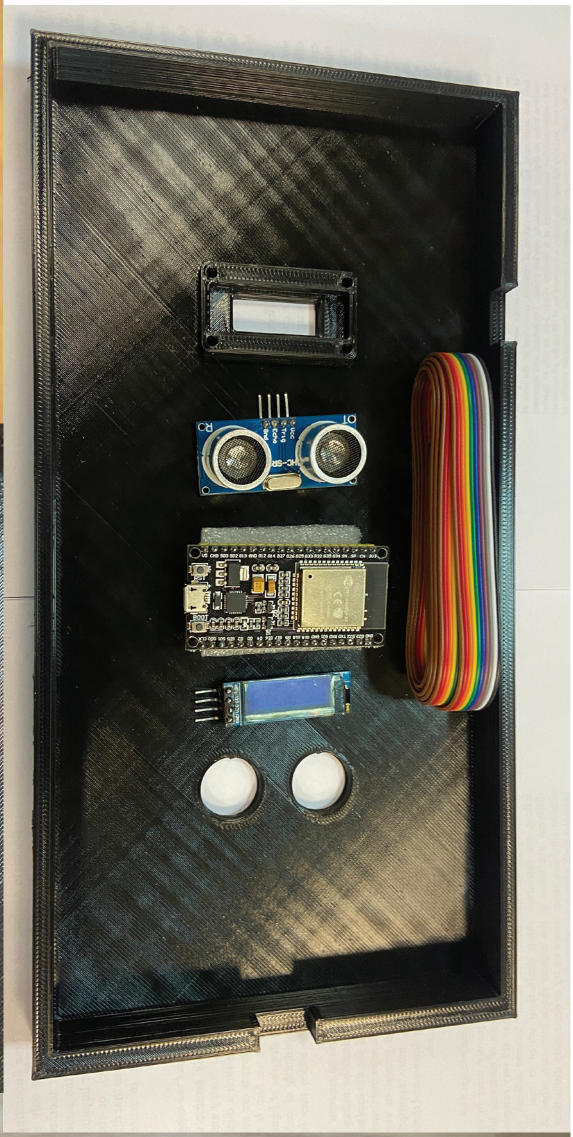
HOMCA





Ender

SHARP
2Y0A02 F 5Z
SHARP
2Y0A21 F 4Y

Ender
Ender

Ender

Ender

Adetec
Adetec
Adetec
MOUNTING TAPE
MICRO USB
MICRO USB



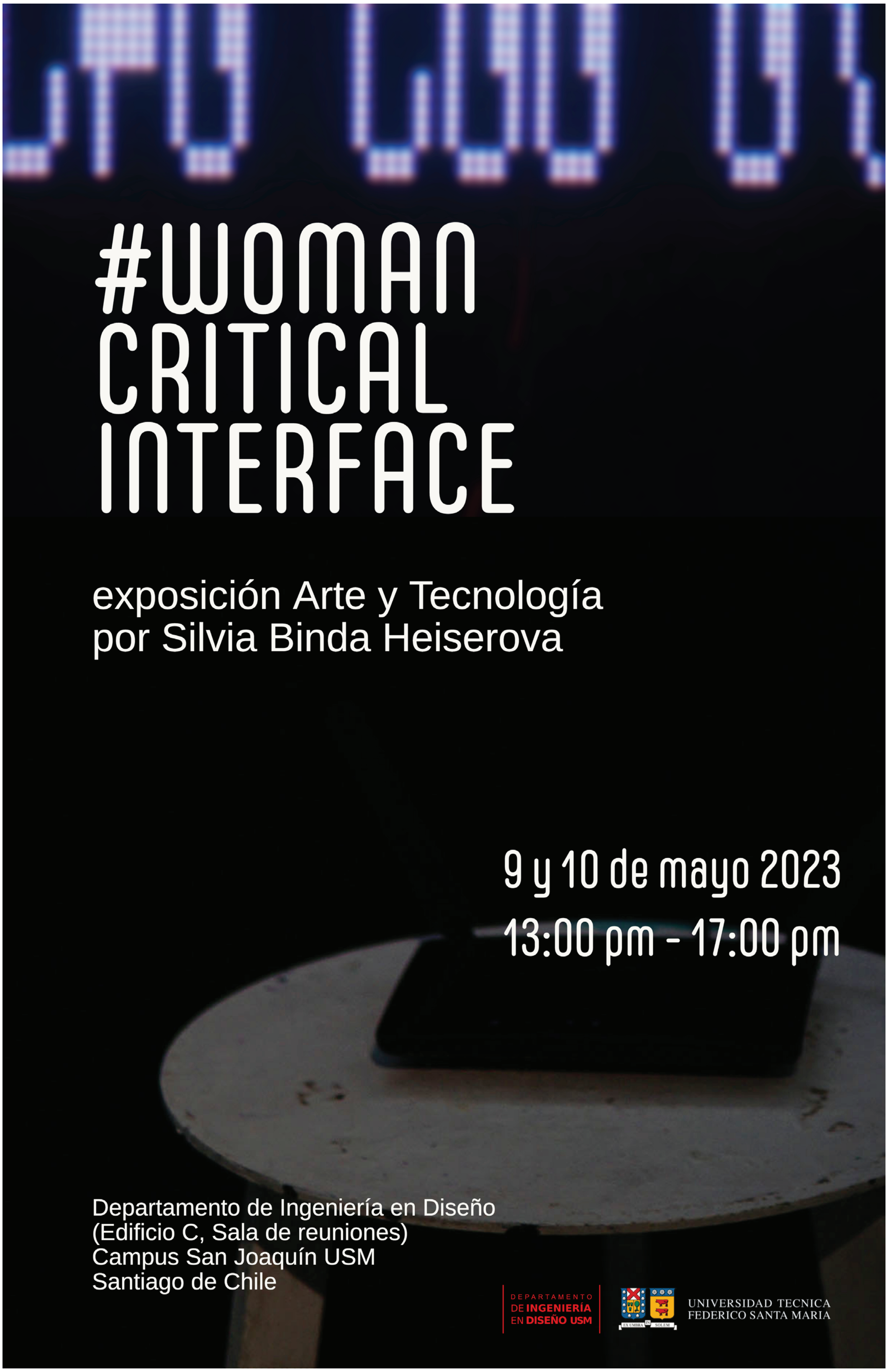


#WOMAN CRITICAL INTERFACE
exposición Arte y Tecnología
por Silvia Binda Heiserova
9 y 10 de mayo 2023
13:00 pm - 17:00 pm
Departamento de Ingeniería en Diseño
(Edificio C, Sala de reuniones)
Campus San Joaquín USM
Santiago de Chile
DEPARTAMENTO DE INGENIERÍA EN DISEÑO USM
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Cartel de la exposición de *#Woman Critical Interface* en Santiago de Chile, mayo de 2023.

#WOMAN CRITICAL INTERFACE
exposición Arte y Tecnología
por Silvia Binda Heiserova
9 y 10 de mayo 2023
13:00 pm - 17:00 pm
Departamento de Ingeniería en Diseño
(Edificio C, Sala de reuniones)
Campus San Joaquín USM
Santiago de Chile
DEPARTAMENTO DE INGENIERÍA EN DISEÑO USM
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Cartel de la exposición de *#Woman Critical Interface* en Santiago de Chile, mayo de 2023.



#WOMAN
CRITICAL
INTERFACE

exposición Arte y Tecnología
por Silvia Binda Heiserova

9 y 10 de mayo 2023
13:00 pm - 17:00 pm

Departamento de Ingeniería en Diseño
(Edificio C, Sala de reuniones)
Campus San Joaquín USM
Santiago de Chile

DEPARTAMENTO
DE INGENIERÍA
EN DISEÑO USM

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Cartel de la exposición de #*Woman Critical Interface* en Santiago de Chile, mayo de 2023.

Nota de prensa publicada en https://usm.cl/noticias/estudiante-de-intercambio-realizo-exposicion-artistica-en-campus-san-joaquin/

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Estudiante de intercambio realizó exposición artística en Campus San Joaquín

Por: Claudia Márquez Rojas, Periodista. Dirección General de Comunicaciones.

4 - julio - 2023

Silvia Binda Heiserova compartió con la comunidad universitaria su muestra #Woman Critical Interface, en el marco de su semestre de intercambio en la Casa de Estudios.

Durante dos jornadas, la estudiante de intercambio entrante Silvia Binda Heiserova expuso su obra #Woman Critical Interface a la comunidad de la Universidad Técnica Federico Santa María en Campus San Joaquín.

La pieza artística forma parte de su proyecto de tesis para el Master en Artes Visuales y Multimedia, que Heiserova cursa en la Universidad Politécnica de Valencia, España, y que le permitió participar de la experiencia de intercambio en la USM durante el primer semestre de este año.

La exposición consiste en una interfaz interactiva, cuyo objetivo es reflexionar críticamente sobre la relación entre la imagen, el lenguaje y el hipertexto desde una perspectiva de género. Tal como explica su autora, "#Woman Critical Interface surge de mi interés de hace muchos años por investigar sobre cuestiones de género y ver las problemáticas sociales desde un punto de vista de género. Mi intención es investigar, por un lado, cómo la mujer es reflejada mediante la tecnología, y también reflexionar sobre el uso del texto relacionado hacia la mujer", explica Heiserova, quien comenta que esta exposición también fue presentada en la Universidad Politécnica de Valencia.

La pieza funciona como un ecosistema digital que se alimenta de datos extraídos de las publicaciones en las redes sociales en tiempo real. Mediante varios *outputs* acústicos y visuales de la interfaz, el usuario experimenta los contenidos originales reinterpretados en nuevos formatos, a lo que se suma las reacciones del sistema frente a cambios producidos en el entorno físico y en el entorno digital.

Diseño, tecnología e inclusión

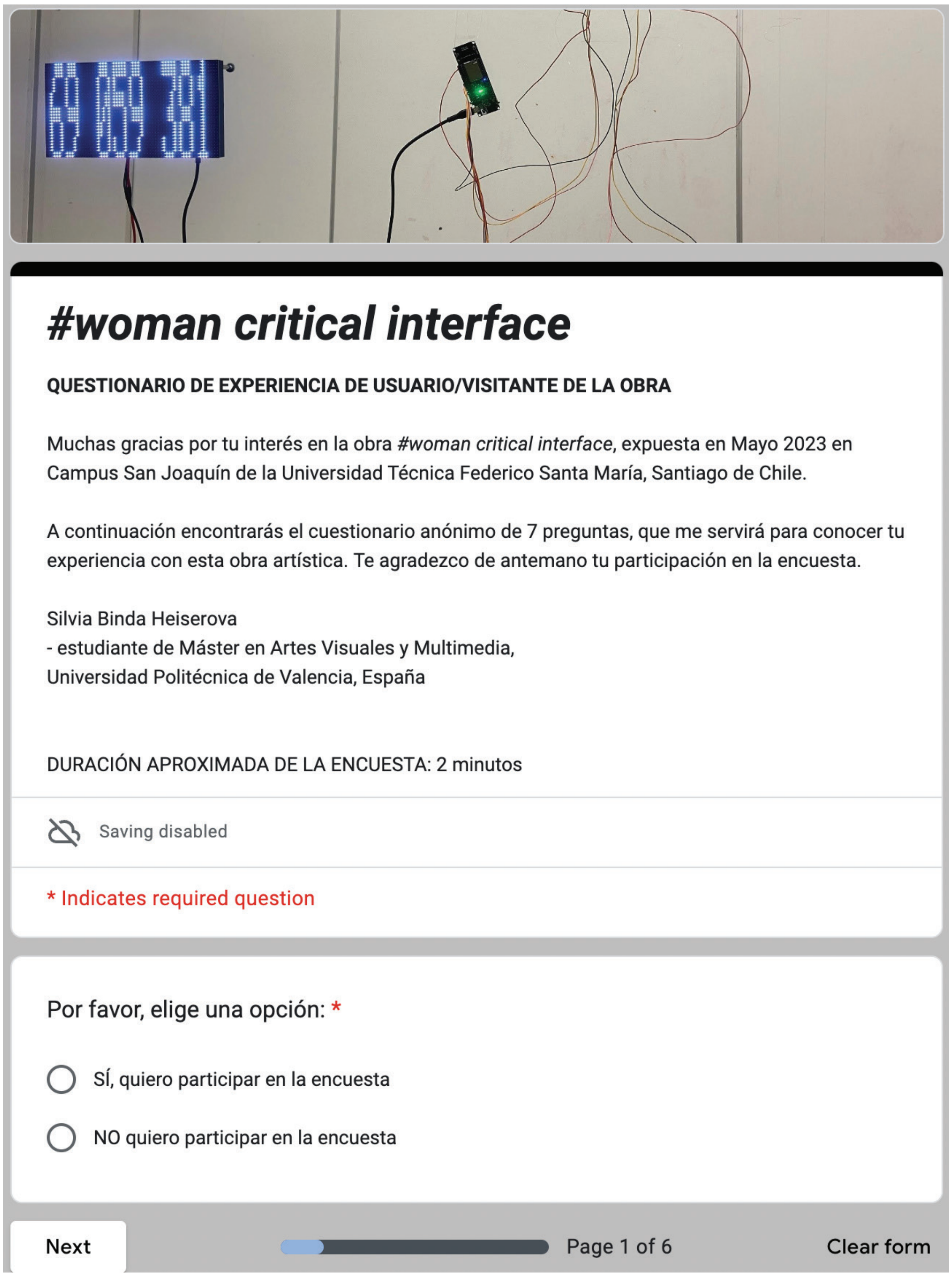
Durante su estadía en la USM, Heiserova ha estado trabajando en su proyecto de tesis con el apoyo de los profesores Mauricio Solar, del Departamento de Informática, y Leonardo Madariaga, del Departamento de Ingeniería en Diseño.

"Los profesores me han ayudado mucho con consejos, comentarios, soluciones y perspectivas nuevas para mi proyecto desde el punto de vista de la ingeniería y la informática, lo que para mí constituye un gran aporte. Poder presentar mi exposición acá también es de gran ayuda, ya que puedo tener un *feedback* diferente y evaluar la experiencia de visitante y usuario. Además, he podido trabajar en el diseño de la pieza con la impresión 3D como componente principal", explica Heiserova.

Leonardo Madariaga, Director del Departamento de Ingeniería en Diseño de la USM, comenta que "nos pareció importante apoyar la instalación de la exposición #Woman Critical Interface dado que combinaba diseño, tecnología y una mirada crítica sobre la perspectiva de género. Cuando alguien conecta una realidad y sus conceptos con una experiencia inmersiva se producen otras ideas en los espectadores y se impulsa un diálogo más fructífero".

La exposición también contó con el apoyo de la Oficina de Asuntos Internacionales (OAI) de la USM. Para Lydia Droegemueller, directora de la OAI, #Woman Critical Interface es un ejemplo del espíritu inclusivo del programa de intercambio. "Como directora de la Oficina de Asuntos Internacionales me llena de orgullo que una estudiante de una de nuestras universidades socias haya elegido venir a la USM para desarrollar este proyecto y compartirlo con nuestra comunidad. Este logro resalta el espíritu inclusivo y enriquecedor de nuestro programa de intercambio, donde las y los estudiantes encuentran su lugar y llevan a cabo actividades académicas y sociales de gran relevancia, incluso si sus programas de estudio en su universidad de origen son distintos a los que ofrecemos en la USM", afirma.

Cuestionario sobre la experiencia del público visitante de la obra *#Woman Critical Interface*



#woman critical interface

QUESTIONARIO DE EXPERIENCIA DE USUARIO/VISITANTE DE LA OBRA

Muchas gracias por tu interés en la obra *#woman critical interface*, expuesta en Mayo 2023 en Campus San Joaquín de la Universidad Técnica Federico Santa María, Santiago de Chile.

A continuación encontrarás el cuestionario anónimo de 7 preguntas, que me servirá para conocer tu experiencia con esta obra artística. Te agradezco de antemano tu participación en la encuesta.

Silvia Binda Heiserova
- estudiante de Máster en Artes Visuales y Multimedia,
Universidad Politécnica de Valencia, España

DURACIÓN APROXIMADA DE LA ENCUESTA: 2 minutos

Saving disabled

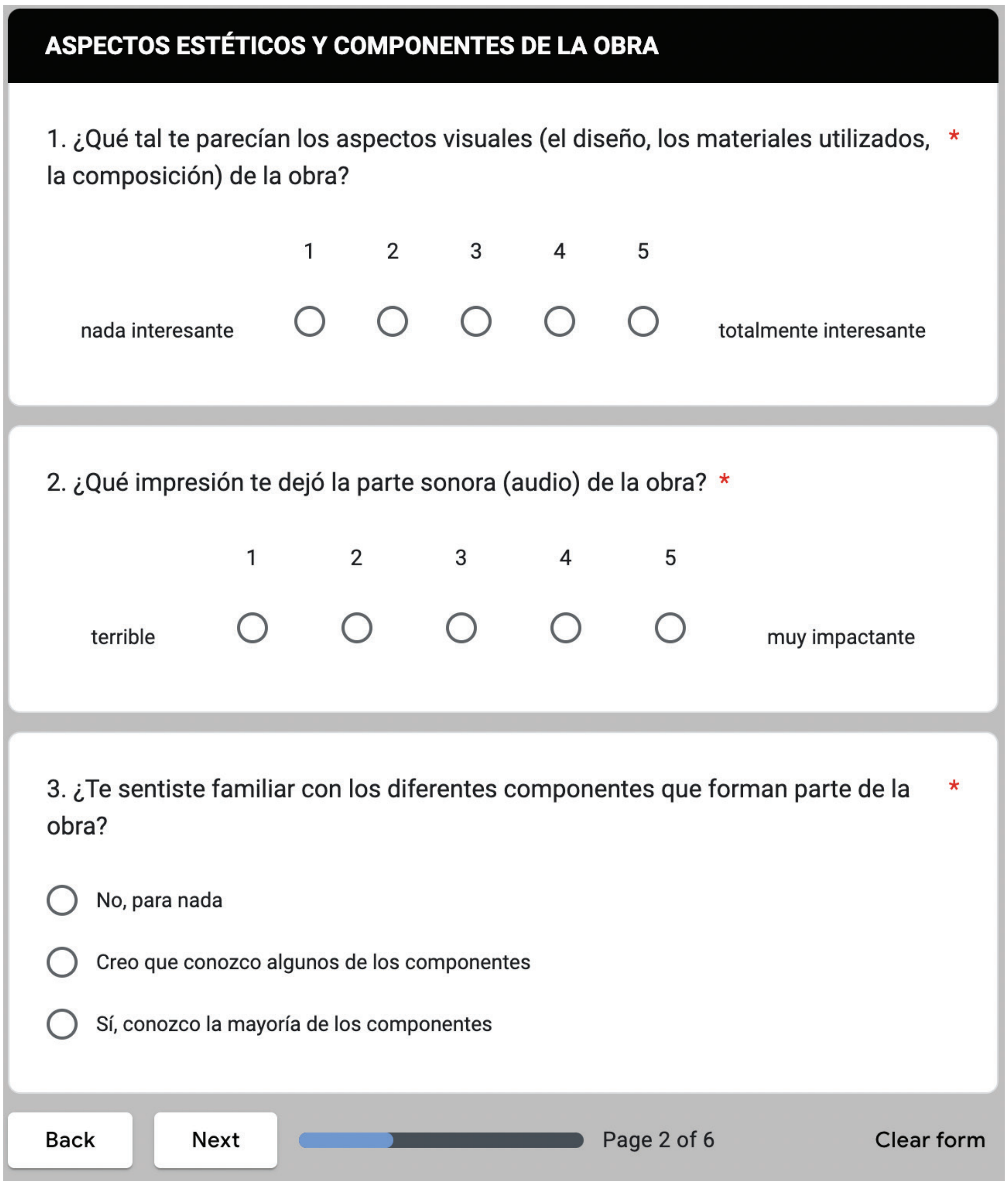
* Indicates required question

Por favor, elige una opción: *

○ SÍ, quiero participar en la encuesta

○ NO quiero participar en la encuesta

Next　Page 1 of 6　Clear form



ASPECTOS ESTÉTICOS Y COMPONENTES DE LA OBRA

1. ¿Qué tal te parecían los aspectos visuales (el diseño, los materiales utilizados, la composición) de la obra? *

	1	2	3	4	5	
nada interesante	○	○	○	○	○	totalmente interesante

2. ¿Qué impresión te dejó la parte sonora (audio) de la obra? *

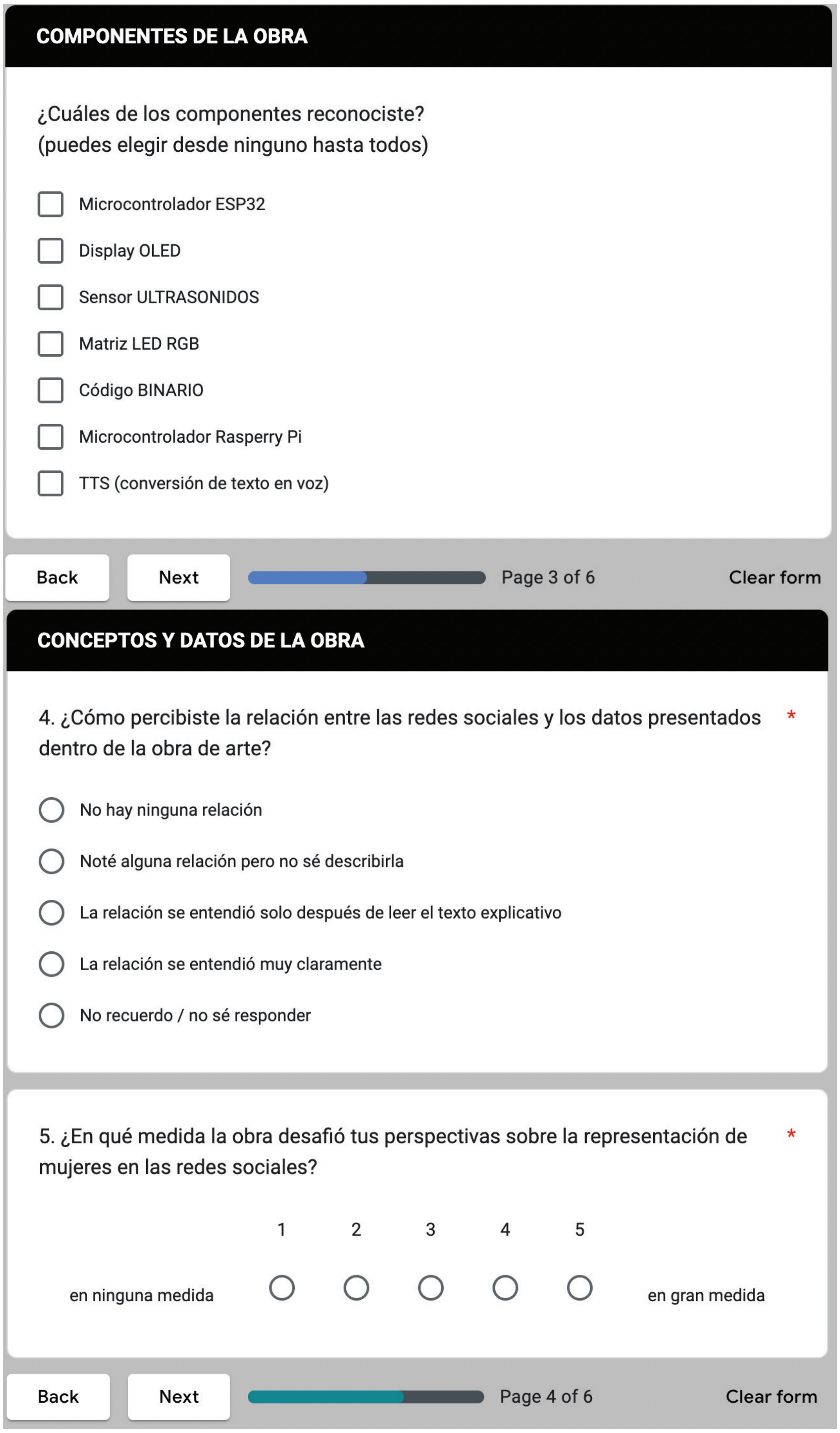
	1	2	3	4	5	
terrible	○	○	○	○	○	muy impactante

3. ¿Te sentiste familiar con los diferentes componentes que forman parte de la obra? *

○ No, para nada

○ Creo que conozco algunos de los componentes

○ Sí, conozco la mayoría de los componentes



COMPONENTES DE LA OBRA

¿Cuáles de los componentes reconociste?
(puedes elegir desde ninguno hasta todos)

- [ ] Microcontrolador ESP32
- [ ] Display OLED
- [ ] Sensor ULTRASONIDOS
- [ ] Matriz LED RGB
- [ ] Código BINARIO
- [ ] Microcontrolador Rasperry Pi
- [ ] TTS (conversión de texto en voz)

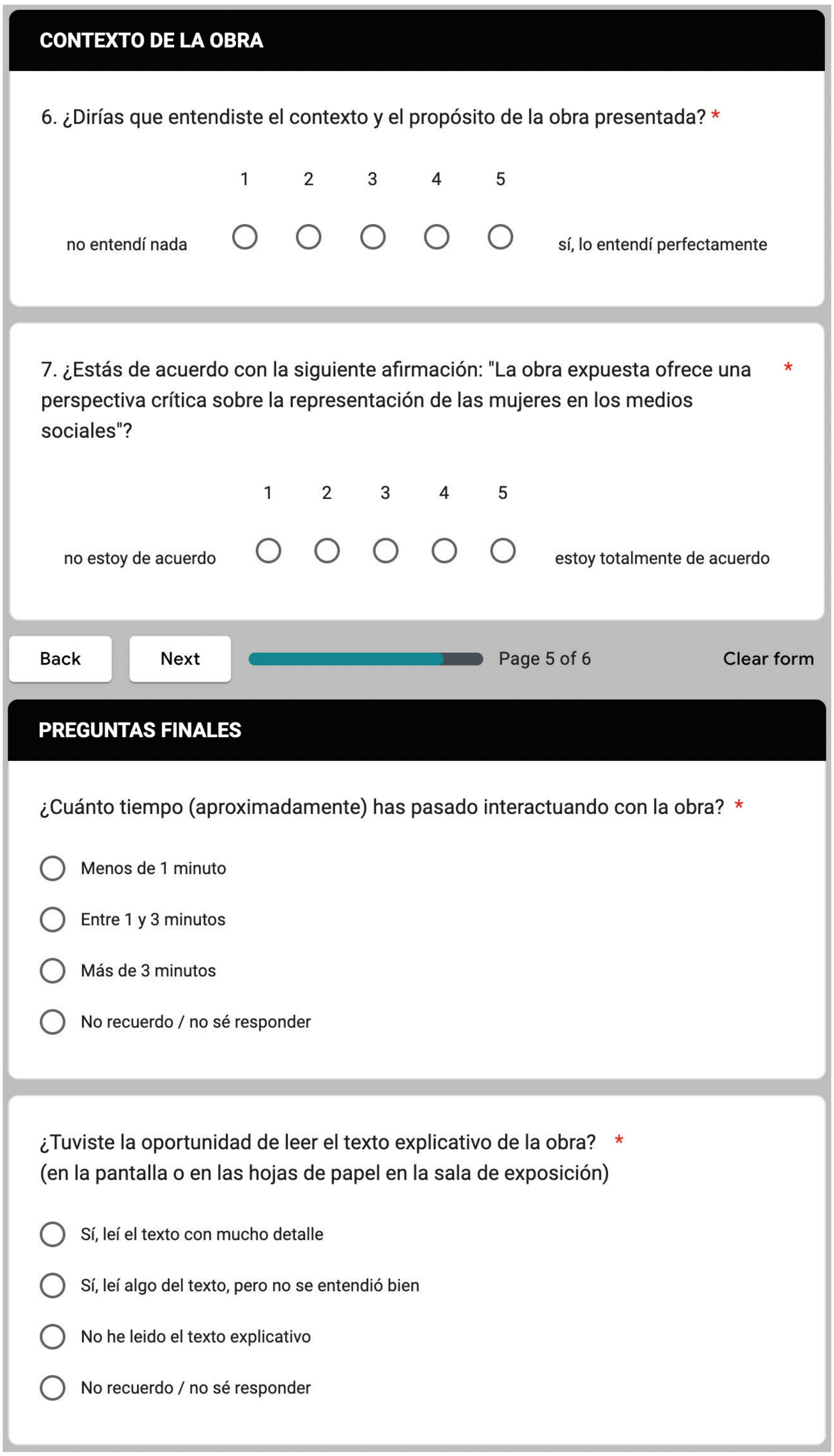
CONCEPTOS Y DATOS DE LA OBRA

4. ¿Cómo percibiste la relación entre las redes sociales y los datos presentados dentro de la obra de arte? *

- ○ No hay ninguna relación
- ○ Noté alguna relación pero no sé describirla
- ○ La relación se entendió solo después de leer el texto explicativo
- ○ La relación se entendió muy claramente
- ○ No recuerdo / no sé responder

5. ¿En qué medida la obra desafió tus perspectivas sobre la representación de mujeres en las redes sociales? *

	1	2	3	4	5	
en ninguna medida	○	○	○	○	○	en gran medida



CONTEXTO DE LA OBRA

6. ¿Dirías que entendiste el contexto y el propósito de la obra presentada? *

	1	2	3	4	5	
no entendí nada	○	○	○	○	○	sí, lo entendí perfectamente

7. ¿Estás de acuerdo con la siguiente afirmación: "La obra expuesta ofrece una perspectiva crítica sobre la representación de las mujeres en los medios sociales"? *

	1	2	3	4	5	
no estoy de acuerdo	○	○	○	○	○	estoy totalmente de acuerdo

PREGUNTAS FINALES

¿Cuánto tiempo (aproximadamente) has pasado interactuando con la obra? *

○ Menos de 1 minuto

○ Entre 1 y 3 minutos

○ Más de 3 minutos

○ No recuerdo / no sé responder

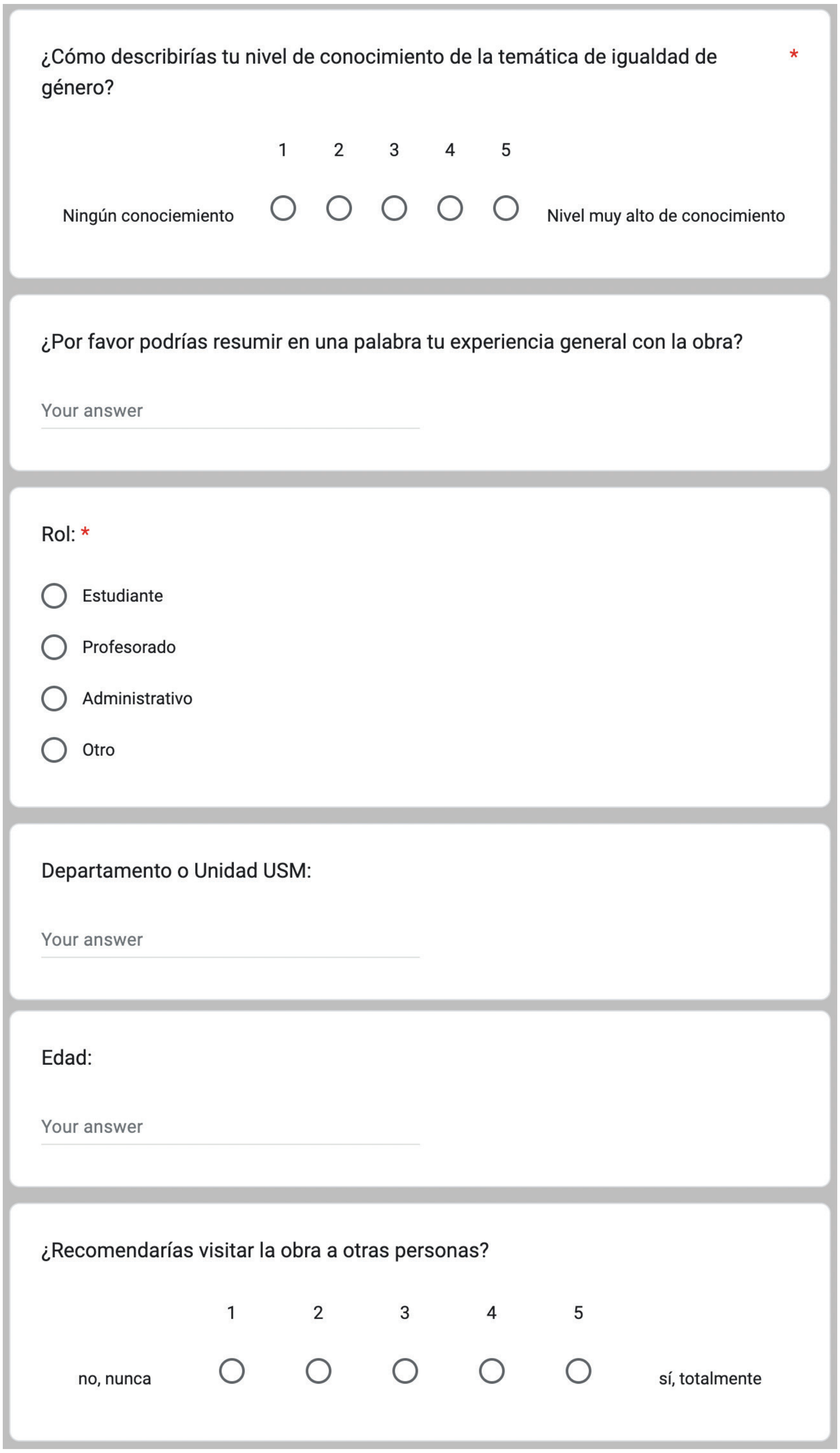
¿Tuviste la oportunidad de leer el texto explicativo de la obra? *
(en la pantalla o en las hojas de papel en la sala de exposición)

○ Sí, leí el texto con mucho detalle

○ Sí, leí algo del texto, pero no se entendió bien

○ No he leido el texto explicativo

○ No recuerdo / no sé responder



¿Cómo describirías tu nivel de conocimiento de la temática de igualdad de género? *

	1	2	3	4	5	
Ningún conociemiento	○	○	○	○	○	Nivel muy alto de conocimiento

¿Por favor podrías resumir en una palabra tu experiencia general con la obra?

Your answer

Rol: *

○ Estudiante

○ Profesorado

○ Administrativo

○ Otro

Departamento o Unidad USM:

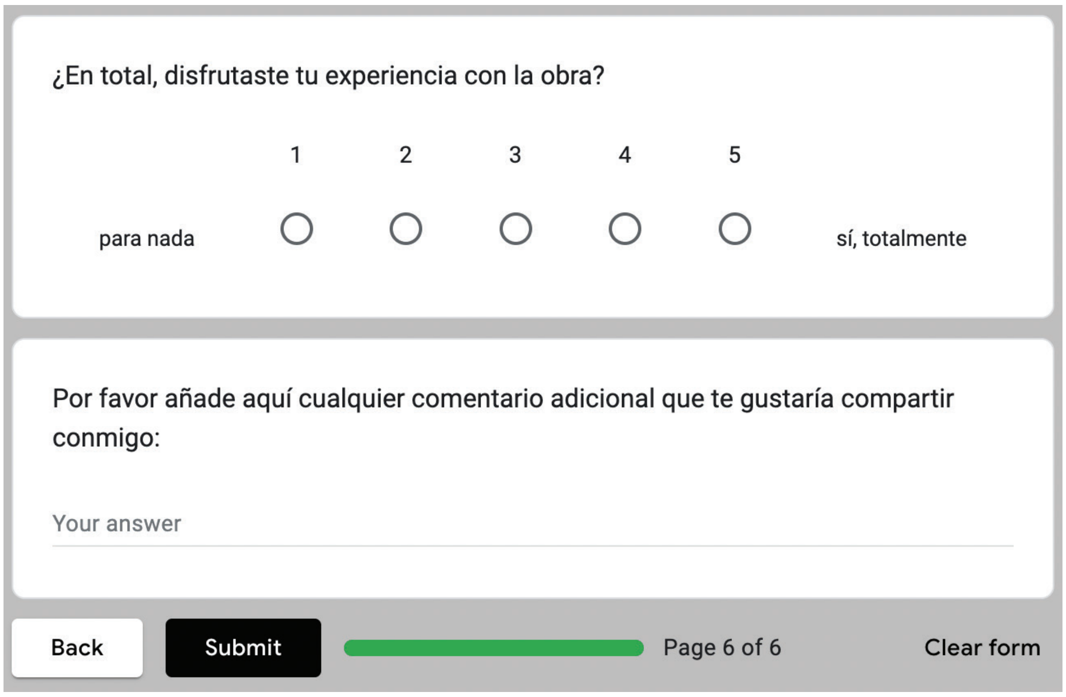
Your answer

Edad:

Your answer

¿Recomendarías visitar la obra a otras personas?

	1	2	3	4	5	
no, nunca	○	○	○	○	○	sí, totalmente


¿En total, disfrutaste tu experiencia con la obra?

	1	2	3	4	5	
para nada	○	○	○	○	○	sí, totalmente

Por favor añade aquí cualquier comentario adicional que te gustaría compartir conmigo:

Your answer

Código QR que lleva a la página web con el cuestionario sobre la experiencia del público visitante de la obra *#Woman Critical Interface*: