Teletubbies Home

Report

IOT Essentials

Pierina Lopez

Academic Year 2021-2022

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

Table of Contents

Teletubbies Home

1. Description

With the Teletubbies home project, it will be posibble to feed the fishes, turn on and off a lamp, fill the aquarium if is needed, mesure the depth, display infotmation in a LCD, displaying information in the UBEAC platform and live Stream the aquarium from anypart of the world.

1.1. Hardware Materials

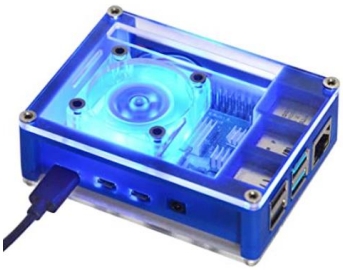
- Rasberry pi 4

- Raspberry pi power supply

- Rpi Camera v2 8 megapixels

- Case for camera

- Case for raspberry



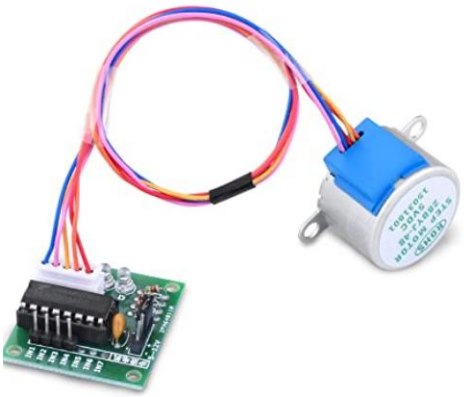
- USB Cable, Tyoe A to type C

- 6 1k ohm resistors

- 3 220 ohm resistors

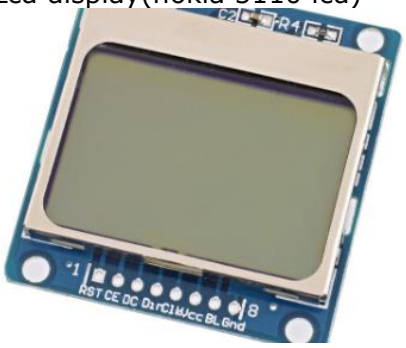
- 1 4 or 2 channel 5v relay board

- 1 stepper motor + driver



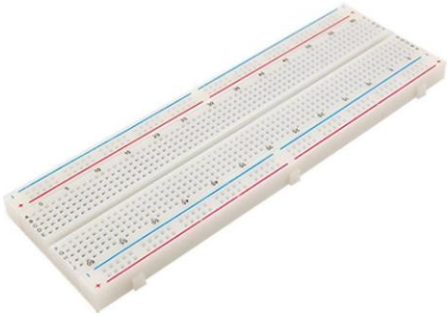
- 1 ultrasonic module

- Lcd display(nokia 5110 lcd)



- Picobbler

- 2 breadboards



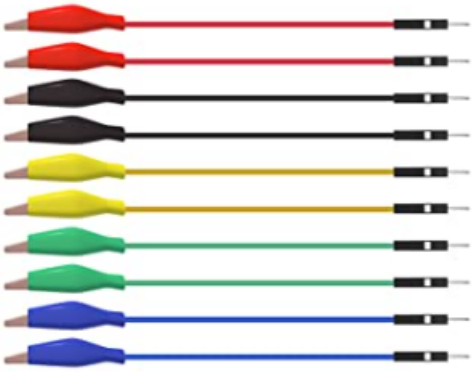
- 16 Gb micro SD

- 4 push buttons

- Wires

- 1 pump 12v/24v

- Alligator clip to breadboard



- 1 power supply module

- 1 red led

- 1 yellow led

- 1 green led

- 1 lamp

- Electrical Insulation Tape

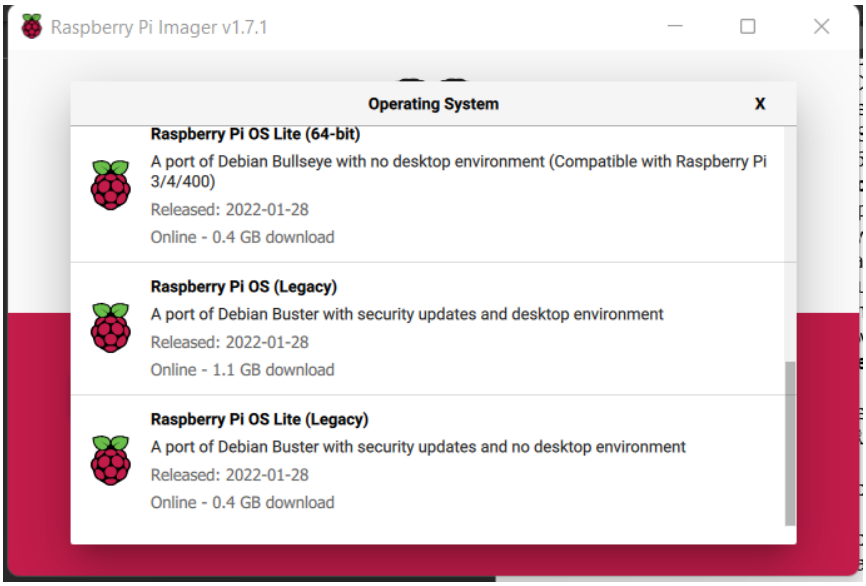
- Screwdriver

1.2. Software and Platforms

- Open CV
- Python 3.7+ & Flask
- Numpy
- Adafruit
- Remote.it

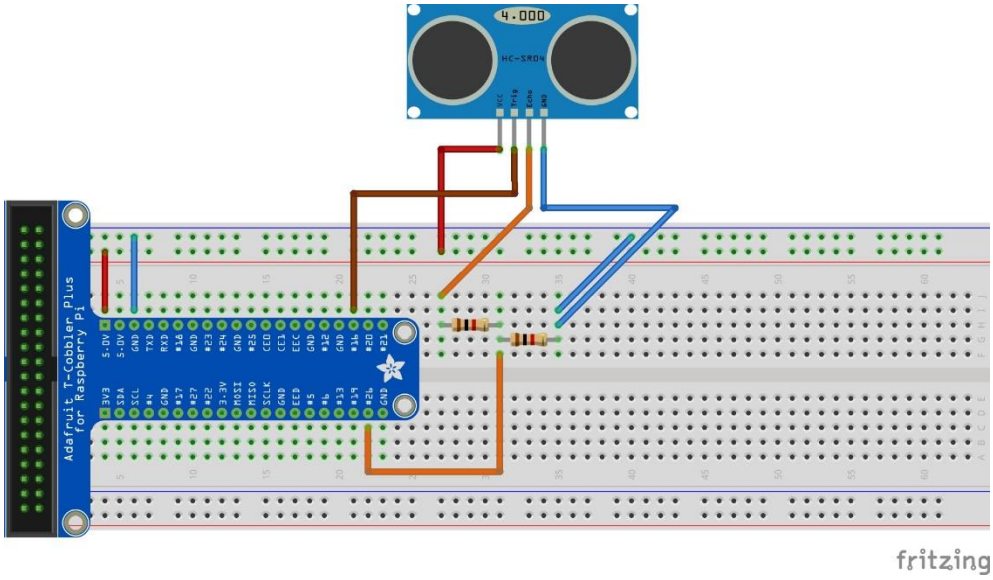
1.3. Setup Procedure:

- Setup Raspberry pi with "Pi Imager" select the Operating System "Raspberry Pi OS (Legacy)"

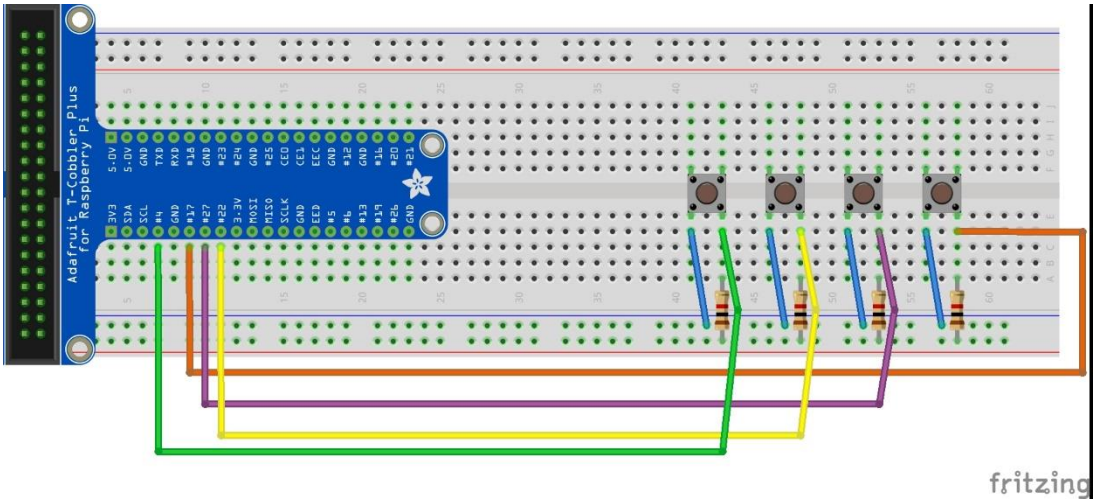


- Connect Rpi Camera to Raspberry pi

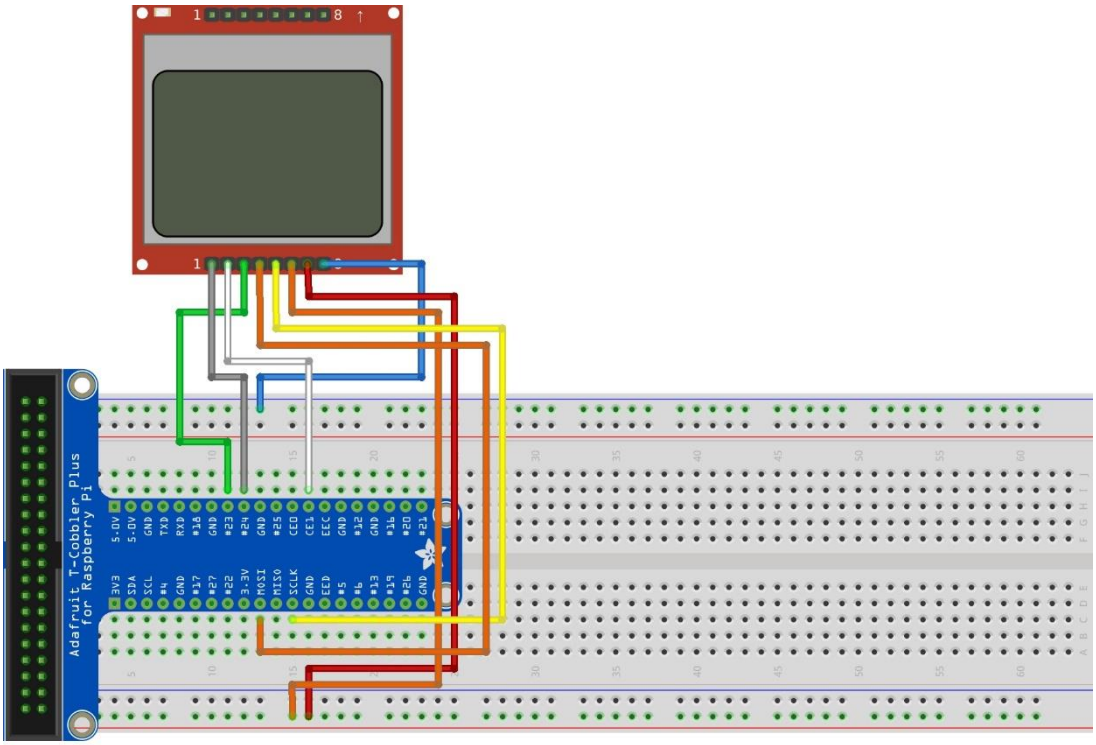
- Connect picobbler to breadboard, connect 2 1k OHM resistor as shown in the graphic:
    - VCC to 5 v
    - Trig to GPIO 16 (BCM)
    - Echo to resitor and to GPIO 26(BCM) as shown in the graphic
    - Gnd to resistor and to Gnd



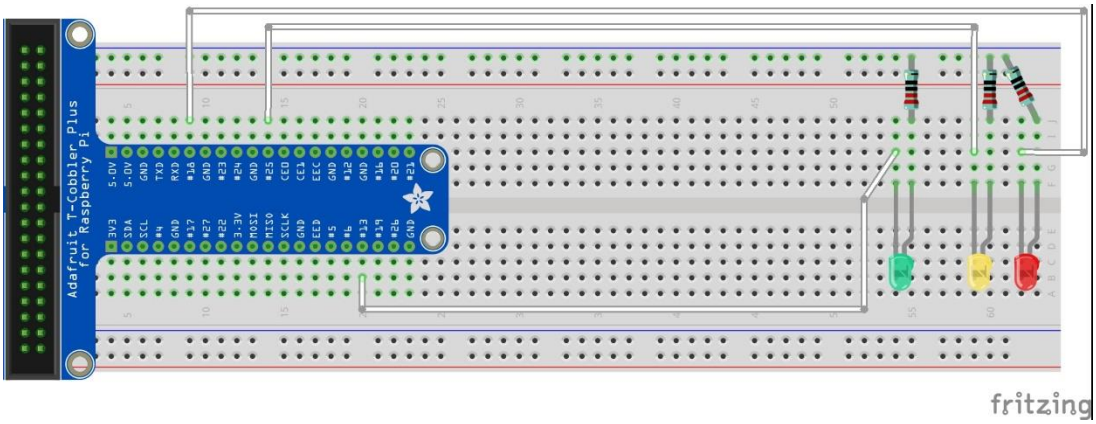
- Connect for buttons and for 1k ohm resistors as shown in the graphic:
    - First button: GPIO 17 / for the pump
    - Second button: GPIO 27/ for the lamp
    - Third button: GPIO 22/ for the feeding to the right
    - Fourth button: GPIO 4/for the feeding to the left



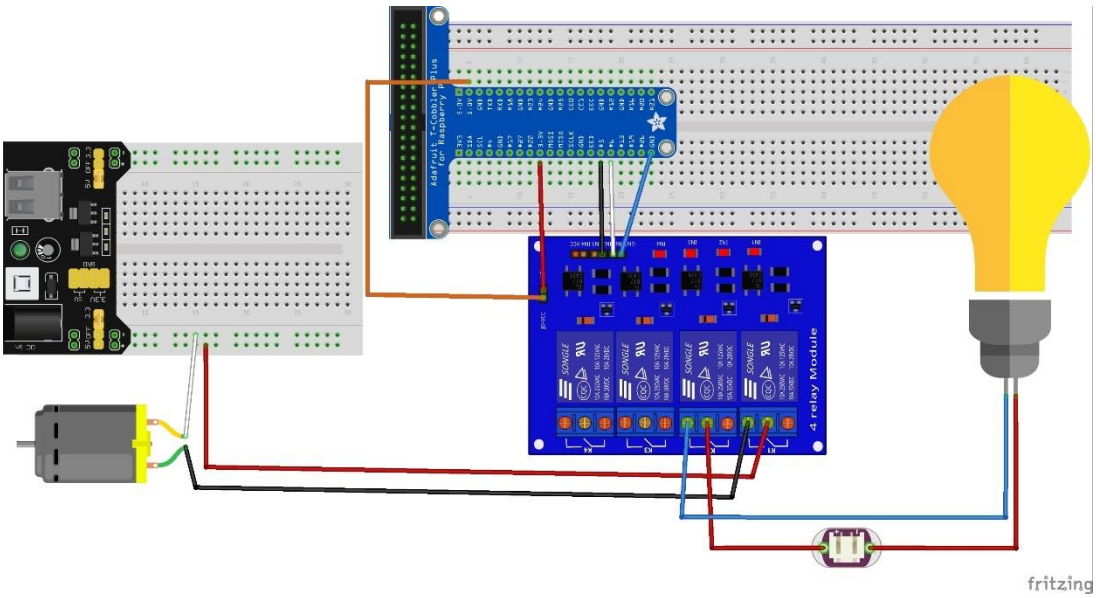
- Connect LCD as shown in graphic:
  - VCC to VCC
  - GROUND TO GROUND
  - DC to Gpio 23
  - Reset to gpio 24
  - Din to Mosi
  - CLK to SCLK
  - CE to CE1



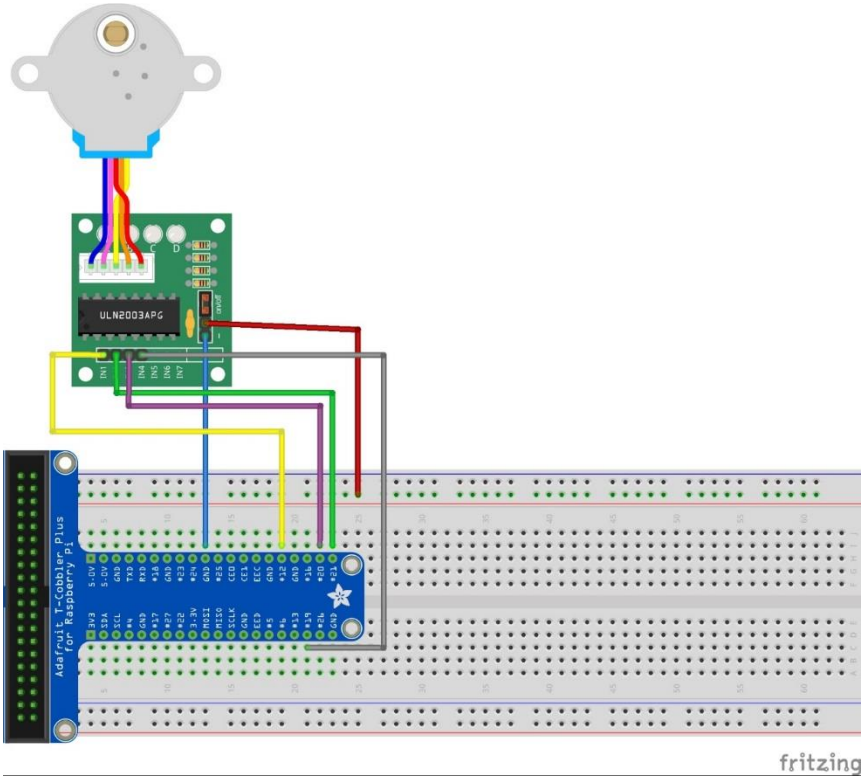
- Connect the led lights as shown in the graphic:
  - Connect 220 resistor to breadboard as shown
  - Green led to GPIO 13
  - Yellow led to GPIO 25
  - Red led to GPIO 18



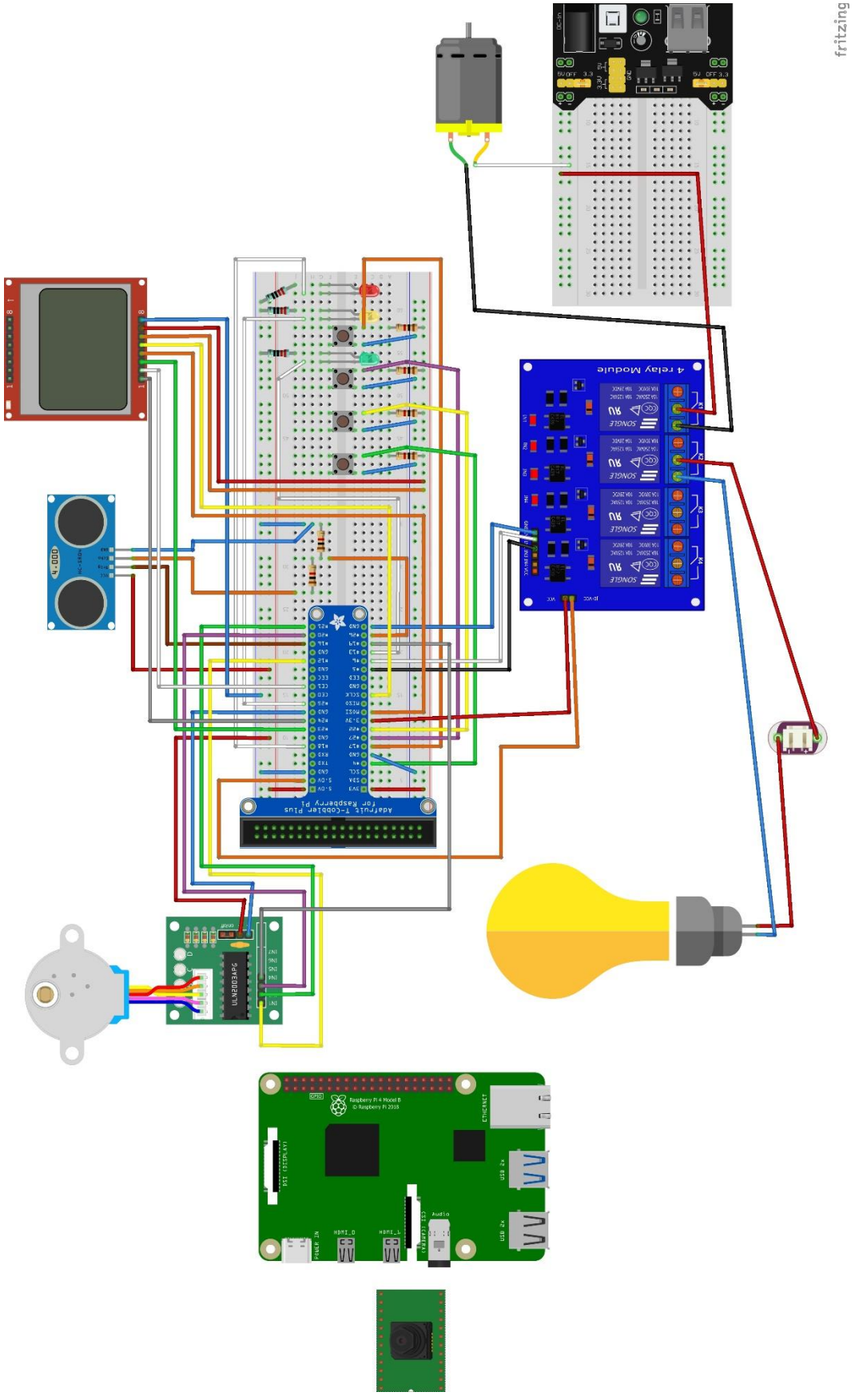
- Connect pump and lamp as shown in the graphic:
  - Relay board first channel to GPIO 6 for the pump
  - Relay board second channel to GPIO 5 for the lamp



- Connect Stepper Motor as shown in the graphic:
  - In1: GPIO 12
  - In2: GPIO 21
  - In3: GPIO 20
  - In4: GPIO 19



- Complete graphic



-

- Install and use Putty for the following part

  - ✓ Upgrade python to 3.8
  - ✓ See what version of python you have with : python -V

    Upgrade with : https://itheo.tech/install-python-38-on-a-raspberry-pi

  Install one at a time:

  - ✓ sudo apt-get update
  - ✓ sudo apt-get upgrade
  - ✓ sudo apt-get install build-essential
  - ✓ sudo apt-get install cmake
  - ✓ sudo apt-get install gfortran
  - ✓ sudo apt-get install git
  - ✓ sudo apt-get install wget
  - ✓ sudo apt-get install curl
  - ✓ sudo apt-get install graphicsmagick
  - ✓ sudo apt-get install libgraphicsmagick1-dev
  - ✓ sudo apt-get install libatlas-base-dev
  - ✓ sudo apt-get install libavcodec-dev
  - ✓ sudo apt-get install libavformat-dev
  - ✓ sudo apt-get install libboost-all-dev
  - ✓ sudo apt-get install libgtk2.0-dev
  - ✓ sudo apt-get install libjpeg-dev
  - ✓ sudo apt-get install liblapack-dev
  - ✓ sudo apt-get install libswscale-dev
  - ✓ sudo apt-get install pkg-config
  - ✓ sudo apt-get install python3-dev
  - ✓ sudo apt-get install python3-numpy
  - ✓ sudo apt-get install python3-pip
  - ✓ sudo apt-get install zip
  - ✓ sudo apt-get clean
- Install the following:
  - ✓ sudo apt-get install python3-picamera
  - ✓ sudo pip3 install --upgrade picamera[array]
- Install supporting dlib libraries:
  - ✓ pip3 install numpy
  - ✓ pip3 install scikit-image
  - ✓ sudo apt-get install python3-scipy
  - ✓ sudo apt-get install libatlas-base-dev
  - ✓ sudo apt-get install libjasper-dev
  - ✓ sudo apt-get install libqtgui4
  - ✓ sudo apt-get install python3-pyqt5
  - ✓ sudo apt install libqt4-test
  - ✓ pip3 install opencv-python==3.4.6.27

2. Remote.it

To be able to see the live stream of the rpi camera, wi will need to set up an account to remote.it

- Install remote.it in to the raspberry pi
  - Sudo apt install remoteit
- Open remote.it in the browser
  - Create an account

        - Register your raspberry
        - Add a new service
            - Service name: camera
            - Service port:5000
            - save

3. **How to use:**
    - To be able to use the program it need to be in the same directory like this:
        - Main directory: TeletubiesWeb
            - Second directory: templates
                - In tamplates: camera.html
            - In TeletubbiesWeb:
                - Allforone.py
                - Lcd.py
                - Main.py
                - Web.py
    - Only need to run Main.py
4. **References:**

- https://www.youtube.com/watch?v=zfBHD4v8hD0
- https://www.youtube.com/watch?v=mQNJpWkdmbc
- https://www.youtube.com/watch?v=DOaDnYj3vfI
- https://www.youtube.com/watch?v=i9mJzdLYsVo

5. **Code**

5.1. Code for funtions: allforone.py

```
#import libraries
import requests
import RPi.GPIO as GPIO
import time
import datetime

#funtion for the pump
def pumpwater():

    #turn off warnings
    GPIO.setwarnings(False)
    #set location mode
    GPIO.setmode(GPIO.BCM)

    #Pump output
    GPIO.setup(6, GPIO.OUT)
    GPIO.output(6, True)

    #sensor output and input
    GPIO.setup(16, GPIO.OUT)
    GPIO.setup(26, GPIO.IN)

    #Pump Button
    GPIO.setup(17, GPIO.IN)
```

```
    #GREEN LED
    GPIO.setup(13, GPIO.OUT)
    #YELLOW LED
    GPIO.setup(25, GPIO.OUT)
    #RED LED
    GPIO.setup(18, GPIO.OUT)


    try:
        #infinite loop
        while True:
            #measure the distance
            #send a 10 µs pulse with the TRIG-pin
            GPIO.output(16, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(16, GPIO.LOW)

            #Loop to record the last timestamp before the signal
reaches the receiver
            while (GPIO.input(26)== GPIO.LOW):
                timestart = time.time()
            #register the last timestamp at which the receiver detects
the signal
            while (GPIO.input(26)== GPIO.HIGH):
                timeend = time.time()
            #calculate time difference between the timestamps
            totaltime = timeend - timestart

            #calculate the difference and multiply with 17000
            depth = totaltime * 17000
            depth = round(depth, 2)

            #create condition, if depth is less than 4 cm and
button  is not pressed
            if depth < 4 and GPIO.input(17)==1:
                #only green led on
                GPIO.output(13, 1)
                GPIO.output(25, 0)
                GPIO.output(18, 0)
                time.sleep(0.1)
                print("Depth is", depth)
                #pump is off
                GPIO.output(6, 1)
                time.sleep(1)

            #condition if depth is less then 4.5 cm and button is not
pressed
            elif depth < 4.5 and GPIO.input(17)==1:
                #only yellow led on
```

```
                GPIO.output(13, 0)
                GPIO.output(25, 1)
                GPIO.output(18, 0)
                time.sleep(0.1)
                print("Depth is", depth)
                #pump off
                GPIO.output(6, 1)
                time.sleep(1)
            #condition if depth is more than 5 cm or button is
pressed
            elif depth > 5 or GPIO.input(17)==0:
                #only red led on
                GPIO.output(13, 0)
                GPIO.output(25, 0)
                GPIO.output(18, 1)
                time.sleep(0.1)
                print("Depth is", depth)
                #pump is on
                GPIO.output(6, 0)
                time.sleep(1)

    finally:
        GPIO.cleanup()
#funtion for feeding
def feeding():
    #turn off warnings
    GPIO.setwarnings(False)

    #set location mode
    GPIO.setmode(GPIO.BCM)

    #button to the left
    bl = 22
    #button to the right
    br = 4
    #set buttons as input
    GPIO.setup(bl, GPIO.IN)
    GPIO.setup(br, GPIO.IN)

    #sequence for button to the right
    CPinR = [12,21,20,19]
    #8 steps sequence
    seq1 = [[1,0,0,1],
            [1,1,0,0],
            [0,1,1,0],
            [0,0,1,1],
            [1,0,0,1],
            [1,1,0,0],
            [0,1,1,0],
```

```
            [0,0,1,1]]

    #sequence for button to the left
    CPinL = [19,20,21,12]
    #8 steps sequence
    seq2 = [[1,0,0,1],
            [1,1,0,0],
            [0,1,1,0],
            [0,0,1,1],
            [1,0,0,1],
            [1,1,0,0],
            [0,1,1,0],
            [0,0,1,1]]

    try:
        while True:
            #if button "bl" is pressed
            if GPIO.input(bl)==0:
                #set pin to out and low
                for pin in CPinL :
                    GPIO.setup(pin, GPIO.OUT)
                    GPIO.output(pin, 0)
                #for loop for the rotation
                for i in range (1):
                    for singlestep in range(8):
                        for pin in range(4):
                                GPIO.output(CPinL[pin],
seq1[halfstep][pin])
                        time.sleep(0.01)
            #if button "br" is pressed
            elif GPIO.input(br)==0:
                #set pin to out and low
                for pin in CPinR:
                    GPIO.setup(pin, GPIO.OUT)
                    GPIO.output(pin, 0)

                #for loop for the rotation
                for i in range (1):
                    for singlestep in range(8):
                        for pin in range(4):
                                GPIO.output(CPinR[pin],
seq2[halfstep][pin])
                        time.sleep(0.01)
            else:
                for pin in CPinL:
                    GPIO.setup(pin, GPIO.OUT)
                    GPIO.output(pin, 0)
    finally:
        GPIO.cleanup()
```

```
#lamp funtion
def lamp():
    #set location mode
    GPIO.setmode(GPIO.BCM)
    #turn off warnings
    GPIO.setwarnings(False)
    #set button as input
    GPIO.setup(27, GPIO.IN)
    #set lamp to output
    GPIO.setup(5, GPIO.OUT)
    #set lamp output to false or low
    GPIO.output(5, False)

    #variable for button state as false
    BS1=False

    try:
        #funtion to recognize if lamp is on or off
        def switch(ev=None):
            nonlocal BS1
            BS1 = not BS1

            if BS1 == True:
                GPIO.output(5, GPIO.HIGH)
            else:
                GPIO.output(5, GPIO.LOW)
        #funtion to change the state of the lamp with button
        def button():
            GPIO.add_event_detect(27, GPIO.FALLING, callback = switch,
bouncetime=300)

        #funtion for the while True
        def wait():
            while True:
                time.sleep(1)
        #starting funtions
        button()
        wait()

    finally:
        GPIO.cleanup()



#funtion to share information to UBEAC
def status():

    #set location mode
    GPIO.setmode(GPIO.BCM)
```

```
    GPIO.setwarnings(False)

    #sensor output and input
    GPIO.setup(16, GPIO.OUT)
    GPIO.setup(26, GPIO.IN)

    #define actuators GPIOs
    lamp = 5
    pump = 6
    #initialize GPIO status variables
    lampSts = 1
    pumpSts = 1

    # Define pins as output
    GPIO.setup(lamp, GPIO.OUT)
    GPIO.setup(pump, GPIO.OUT)
    # turn pins OFF
    GPIO.output(lamp, GPIO.HIGH)
    GPIO.output(pump, GPIO.HIGH)

    #SET URL AND UID PROVIDE FROM UBEAC
    url = "http://itproject.hub.ubeac.io/iotpierina"
    uid = "iotPierina"

    def readdepth():
        #measure the distance
        #send a 10 µs pulse with the TRIG-pin
        GPIO.output(16, GPIO.HIGH)
        time.sleep(0.00001)
        GPIO.output(16, GPIO.LOW)

        #Loop to record the last timestamp before the signal reaches
the receiver
        while (GPIO.input(26)== GPIO.LOW):
            timestart = time.time()
        #register the last timestamp at which the receiver detects the
signal
        while (GPIO.input(26)== GPIO.HIGH):
            timeend = time.time()
        #calculate time difference between the timestamps
        totaltime = timeend - timestart
        #calculate the difference and multiply with 17000
        depth = totaltime * 17000
        depth = round(depth, 2)

        return depth

    def readlamp():
        while True:
```

```
            # Read Sensors Status
            lampSts = GPIO.input(lamp)
            return lampSts

    def readpump():
        while True:
            # Read Sensors Status
            pumpSts = GPIO.input(pump)
            return pumpSts
    #endless loop for the reading data
    while True:
        pumpdata=readpump()
        depthdata=readdepth()
        lampdata=readlamp()
        data={
            "id": uid,
            "sensors":[
                {
                'id': 'lamp Status',
                'data': lampdata
                },
                {
                'id': 'Depth',
                'data': depthdata
                },
                {
                'id': 'pump Status',
                'data': pumpdata
                }]
        }
        r = requests.post(url, verify=False, json = data)
        time.sleep(1)
```

5.2. Display code: lcd.py

```
def lcd():
    #import libraries
    import time
    import cgitb
    from os import read #cgitb.enable()
    import spidev
    import busio
    import digitalio
    import board
    import adafruit_pcd8544
    from adafruit_bus_device.spi_device import SPIDevice
    from PIL import Image
```

```
    from PIL import ImageDraw
    from PIL import ImageFont
    import datetime
    import RPi.GPIO as GPIO

    #set location mode
    GPIO.setmode(GPIO.BCM)
    #turn off warnings
    GPIO.setwarnings(False)
    #variable for the lamp pin
    lamp = 5
    #set lamp pin to output and high
    GPIO.setup(lamp, GPIO.OUT)
    GPIO.output(lamp, GPIO.HIGH)

    #initialize GPIO status variables
    lampSts = 1

    # Initialize SPI bus
    spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

    cs0 = digitalio.DigitalInOut(board.CE0)

    #funtion for countdown to feeding time
    def countdown():

        def dateDiffInSeconds(date1, date2):
          timedelta = date2 - date1
          return timedelta.seconds

        def daysHoursMinutesSecondsFromSeconds(seconds):
            minutes, seconds = divmod(seconds, 60)
            hours, minutes = divmod(minutes, 60)
            days, hours = divmod(hours, 24)
            return (days, hours, minutes, seconds)

        req = datetime.datetime.strptime('2025-05-08 17:03:30', '%Y-
%m-%d %H:%M:%S')
        now = datetime.datetime.now()

        while req>now:
            countdown = "%dd %dh %dm %ds" %
daysHoursMinutesSecondsFromSeconds(dateDiffInSeconds(now, req))
            return (countdown)
            sleep(1)
            now = datetime.now()

            if countdown == "0d 0h 0m 0s":
                return("Fedding time")
```

```
                sleep(10)
    #funtion to read the depth
    def readdepth():
        #sensor output and input
        GPIO.setup(16, GPIO.OUT)
        GPIO.setup(26, GPIO.IN)
        #measure the distance
        GPIO.output(16, GPIO.HIGH)
        time.sleep(0.00001)
        GPIO.output(16, GPIO.LOW)

        while (GPIO.input(26)== GPIO.LOW):
            timestart = time.time()

        while (GPIO.input(26)== GPIO.HIGH):
            timeend = time.time()

        totaltime = timeend - timestart

        depth = totaltime * 17000
        depth = round(depth, 2)

        return depth
    #funtion to read the status of lamp
    def readlamp():
        lampSts = GPIO.input(lamp)
        return lampSts

    #funtion to display current time
    def daate():
        return datetime.datetime.now().strftime('%H:%M:%S')

    # Initialize display
    dc = digitalio.DigitalInOut(board.D23) # data/command
    cs1 = digitalio.DigitalInOut(board.CE1) # chip select CE1 for
display
    reset = digitalio.DigitalInOut(board.D24) # reset
    display = adafruit_pcd8544.PCD8544(spi, dc, cs1, reset, baudrate=
1000000)
    display.bias = 4
    display.contrast = 60
    display.invert = True

    # Clear the display. Always call show after changing pixels to
make the display update visible!
    display.fill(0)
    display.show()

    # Load default font.
```

```
    #font = ImageFont.load_default()
    font =
ImageFont.truetype("/usr/share/fonts/truetype/freefont/FreeSansBold.tt
f", 10)

    # Get drawing object to draw on image
    image = Image.new('1', (display.width, display.height))
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.
    draw.rectangle((0, 0, display.width, display.height), outline=255,
fill=255)

    #loop for displaying information
    while True:
        display.fill(0)
        display.show()
        draw.rectangle((0, 0, display.width, display.height),
outline=255, fill=255)

        draw.text((1,1), "Time: " + str(daate()), font=font)
        draw.text((1,9), "Lamp: " + str(readlamp()), font=font)
        draw.text((1,18), "Depth: " + str(readdepth()), font=font)
        draw.text((1,27), "Next fedding:", font=font)
        draw.text((1,36), str(countdown()), font=font)
        display.image(image)
        display.show()
        time.sleep(1)
```

5.3. Code for the web page: web.py

```
#import libraries
import cv2
import numpy
from flask import Flask, render_template, Response,
stream_with_context, request, redirect, url_for
import time
import RPi.GPIO as GPIO
import datetime
import psutil
import os

#set location mode
GPIO.setmode(GPIO.BCM)
#turn off warnings
GPIO.setwarnings(False)
#variable for the lamp pin
lamp = 5
```

```
#set lamp pin to output and high
GPIO.setup(lamp, GPIO.OUT)
GPIO.output(lamp, GPIO.LOW)
#initialize GPIO status variables
lampSts = 1

#starting the video
video = cv2.VideoCapture(0)
app = Flask('__name__')

#funtion for the video streaming
def video_stream():
    while True:
        ret, frame = video.read()
        if not ret:
            break;
        else:
            ret, buffer = cv2.imencode('.jpeg',frame)
            frame = buffer.tobytes()
            yield (b' --frame\r\n' b'Content-type:
imgae/jpeg\r\n\r\n' + frame +b'\r\n')

#set route for funtion to show the current time
@app.route("/camera")
def timeserver():
    now = datetime.datetime.now()
    timeString = now.strftime("%Y-%m-%d %H:%M")
    templateDate = {'time' : timeString}
    return render_template('camera.html',**templateDate)

#set route for funtion to read the light status
@app.route("/camera")
def lights():
    lampSts = GPIO.input(lamp)
    templateData = {
        'lamp' : lampSts
        }
    return render_template('camera.html', **templateData)

#set route for funtion of buttons in webpage to turn on and off the
lamp
@app.route("/<deviceName>/<action>")
def action(deviceName, action):
    if deviceName == 'lamp':
        actuator = lamp
    if action == "on":
        GPIO.output(actuator, GPIO.LOW)
        time.sleep(900)
        GPIO.output(actuator, GPIO.HIGH)
```

```
    if action == "off":
        GPIO.output(actuator, GPIO.HIGH)

    lampSts = GPIO.input(lamp)
    templateData = {
        'lamp'   : lampSts
        }
    return render_template('camera.html', **templateData)
#route for the template
@app.route('/camera')
def camera():
    return render_template('camera.html')
#route to show streaming video
@app.route('/video_feed')
def video_feed():
    return Response(video_stream(), mimetype='multipart/x-mixed-
replace; boundary=frame')

#set rule to connect to webpage
app.run(host='0.0.0.0', port='5000', debug=False)
```

5.4. Template for web page: camera.html

```
<html>
 <head>
   <meta name="viewport" content="width=device-width, initial-
scale=1">
   <style>
      body {background:black;color: white}
      img {display: block;margin-left: auto;margin-right: auto;}
      h1 {text-align: center;}
      h2 {text-align: center;}
      button {font: bold 15px Arial;text-decoration:
none;background: darkcyan;color: white;padding: 2px 6px 2px 6px;
              border-top: 1px solid #CCCCCC;border-right: 1px solid
#333333;border-bottom: 1px solid #333333;
              border-left: 1px solid #CCCCCC;
              }
      button {background: darkcyan;color:white}
    </style>
  </head>
  <body>
     <h1>Teletubbies Home</h1>
     <h2>Surveillance Camera</h2>
     <h2>Server Time: {{ time }}</h2>
     <h3>
         <a href="/lamp/on"><button>TURN ON LAMP</button></a>
         <a href="/lamp/off"><button>TURN OFF LAMP</button></a>
```

```
    </h3>
    <img id="bg" src="{{ url_for('video_feed') }}"
style="width:88%;">
  </body>
</html>
```

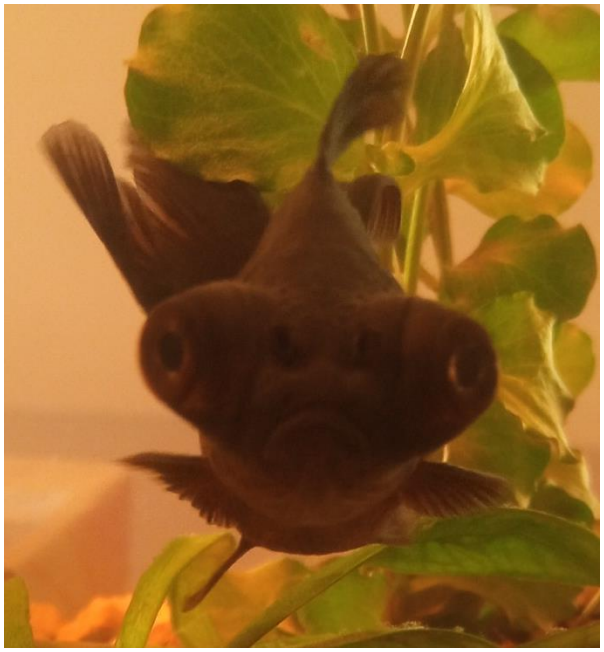
5.5. Code for main program: main.py

```
#import libraries
from allforone import pumpwater,feeding,lamp,status
from multiprocessing import Process
from lcd import lcd
from web import webtst


if __name__ == "__main__":

    #set variables for funtions
    w = Process(target=pumpwater)
    f = Process(target=feeding)
    l = Process(target=lamp)
    s = Process(target=status)
    d = Process(target=lcd)
    t = Process(target=webtst)

    #starting funtions
    t.start()
    w.start()
    f.start()
    l.start()
    s.start()
    d.start()

    #joining the funtions
    t.join()
    w.join()
    f.join()
    l.join()
    s.join()
    d.join()
```

6. Additional Set up

For this project I bought 4 golden fishes and named them as the Teletubbies characters:

- Po

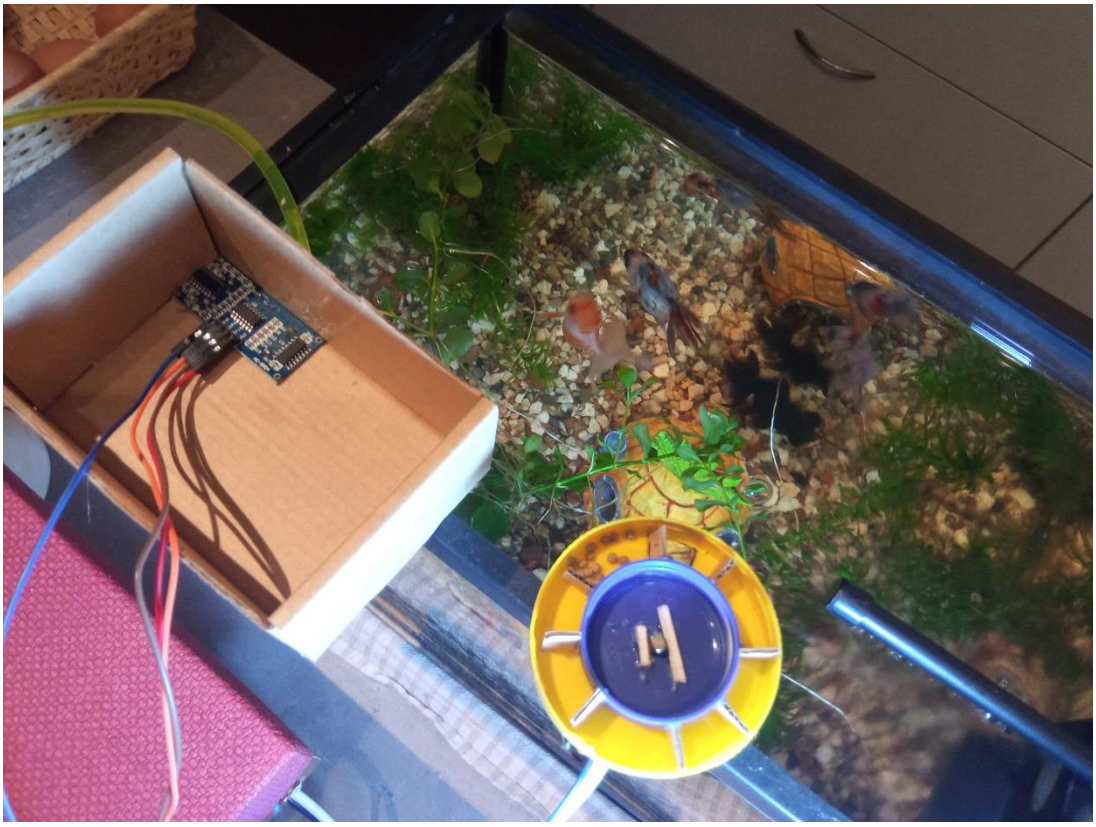


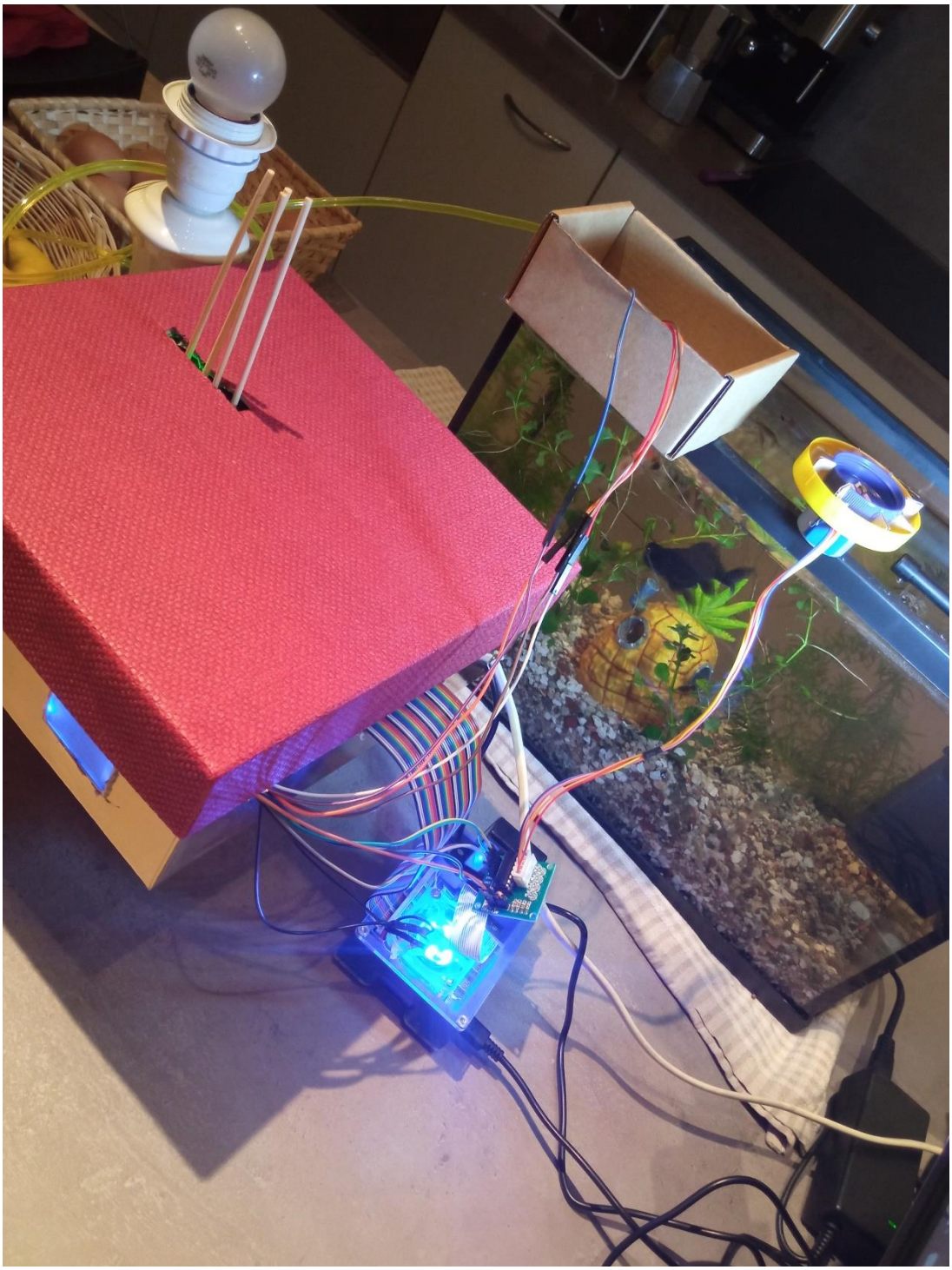
- Tinky-Winky

- Laa-laa

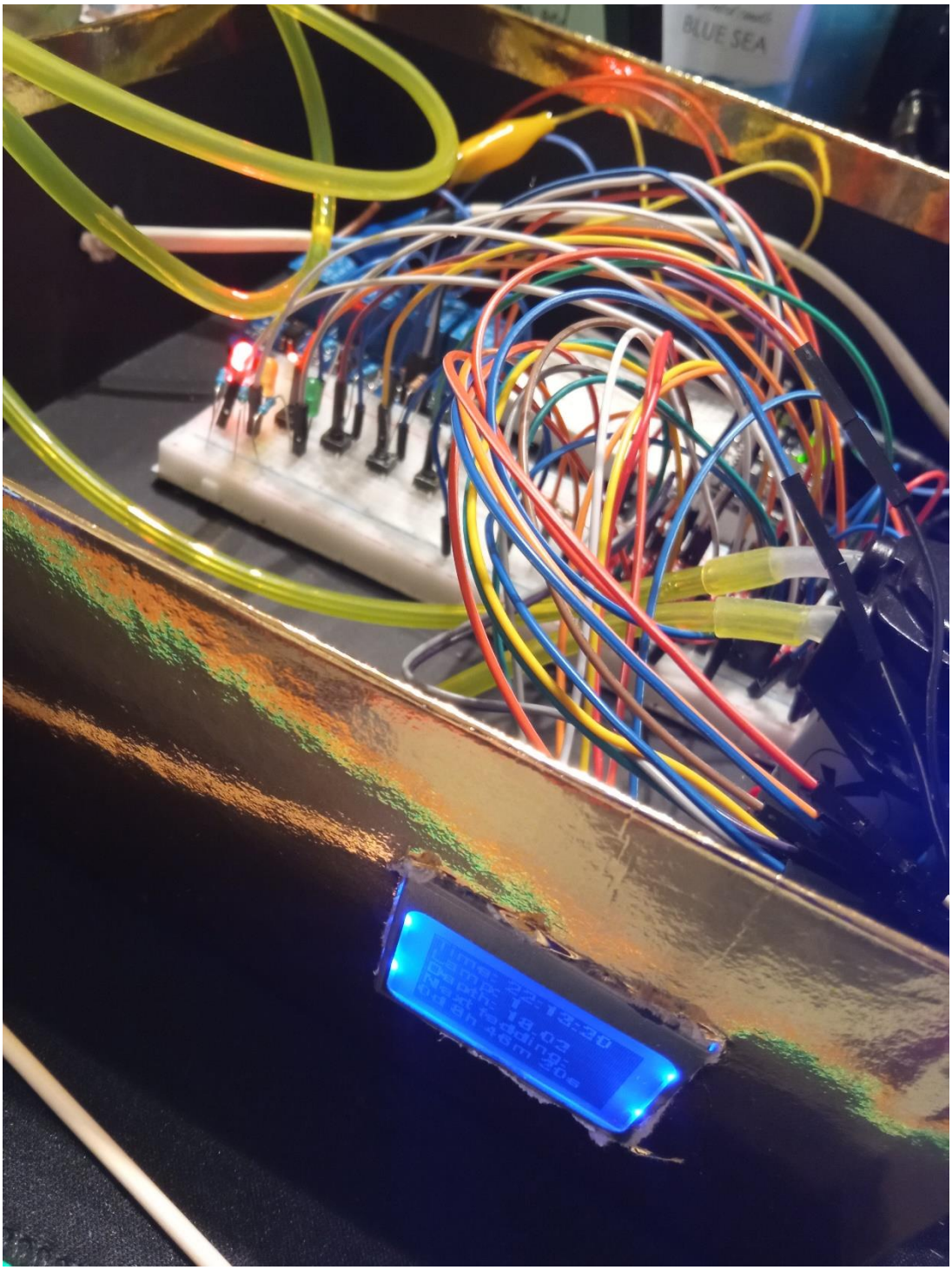
- Dipsy

7. Pictures of project






Time: 22:13:23
Lamp: 1
Depth: 18.02
Next feedding:
0d 8h 46m 36s


BLUE SEA