

HOME SECURITY / DEFENSE SYSTEM

# IOT Group Project

GROUP 10

# Table of Contents

<b>1. TEAM</b>	<b>4</b>
1.1. Team composition	4
1.2. Responsibilities	4
<b>2. PROJECT INTRODUCTION</b>	<b>5</b>
2.1. Project Definition	5
2.1.1. How will we meet the evaluation criteria?	5
2.2. Description	5
2.3. Information value loop	6
2.3.1. Current state and behavior	6
2.3.2. Read out with sensors	6
2.3.3. Communicate through embedded communication protocols	6
2.3.4. Transport data to Embedded platforms	7
2.3.5. Communicating through communication protocols	7
2.3.6. Transporting data to cloud platforms	7
2.3.7. Supporting Analytical and intelligence tools	7
2.3.8. Value driven actions	7
2.4. Schematic	8
2.5. Pictures	9
<b>3. PROJECT CODE</b>	<b>11</b>
3.1. Code printout	11
3.1.1. Raspberry PI Pico	11
3.1.2. OrangePI 3 LTS	17
3.2. Detailed Code explanation	24
3.2.1. Raspberry PI Pico	24
3.2.2. OrangePI 3 LTS	32
<b>4. HARDWARE EXPLANATION</b>	<b>45</b>
4.1. List of components	45
4.1.1. Standard components	45
4.1.2. Extra components	45
4.2. Components	47
4.2.1. Jumper cables	47
4.2.2. (1x) Raspberry PI pico	47
4.2.3. (1x) OrangePI 3 LTS	50
4.2.4. (1x) HC-SR04 Sensor (Distance)	52
4.2.5. (2x) 220 ohm resistor	55
4.2.6. (1x) LCD 5110	56
4.2.7. (1x) Watergun	63
4.2.8. (5x) L298N Motor Driver	64
4.2.9. (1x) Smoke machine	70
4.2.10. (2x) 12V Battery holders	73
4.2.11. (5x) Nema 17	74

4.2.12. (4x) Shaft coupler	77
<b>5. PHYSICAL STRUCTURE</b>	<b>78</b>
5.1. Software to make parts	78
5.1.1. FreeCAD	78
5.2. 3D Printed components	78
5.2.1. Main Gun mount	78
5.2.2. Gun Barrel mount	78
5.2.3. Camera and distance sensor mount	79
5.2.4. Gun Top	79
5.2.5. Gun bottom	80
<b>6. VIDEO</b>	<b>81</b>
<b>7. SELF EVALUATION</b>	<b>82</b>
<b>8. REFERENCES</b>	<b>83</b>
8.1. DataSheets / User manuals	83
8.2. Schematics	83
8.3. Other	83

# 1. Team

## 1.1. Team composition

- Alex van Poppel
- Gabriela Betancourth Rodrigues
- Rik Dekkers
- Tijs Kanter

## 1.2. Responsibilities

			
<b>TEAM LEADER</b>	<b>DOCUMENTATION</b>	<b>HARDWARE</b>	<b>SOFTWARE</b>
<i>Tijs Kanter</i>	<i>Rik Dekkers</i>	<i>Alex van Poppel</i>	<i>Gabriela Betancourth Rodrigues</i>



## 2. Project Introduction

### 2.1. Project Definition

#### 2.1.1. How will we meet the evaluation criteria?

Sensors, communication	Digital input and output	Camera input
	Analog input	Data from light distance sensor
	I <sup>2</sup> C & SPI	I <sup>2</sup> C – camera signal   SPI – LCD screen
Embedded platforms	Orange Pi	Central processor, running AI and communicate
	Raspberry Pi Pico	On gun, receive all data + send to OrangePI
Cloud & communication	MQTT	Collect AI data
	Online dashboard	uBEAC combine data from different sensors
Analyze	AI	Recognize where people are
Act	Stepper motor	Rotates gun into position

### 2.2. Description

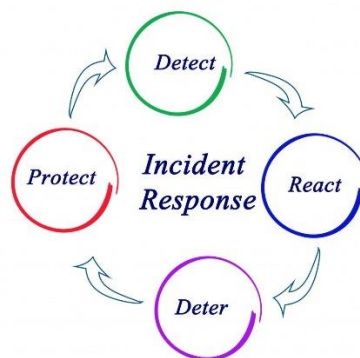
We live in a world where dangers are lurking everywhere...

In a world like this, we have to be able to keep ourselves safe, while also trying to make sure we don't have to dedicate our entire lives preparing for the worst. Wouldn't it be amazing if we could use today's technology to make a machine take a large part of the defensive measures of our backs, so we can focus on more important things?

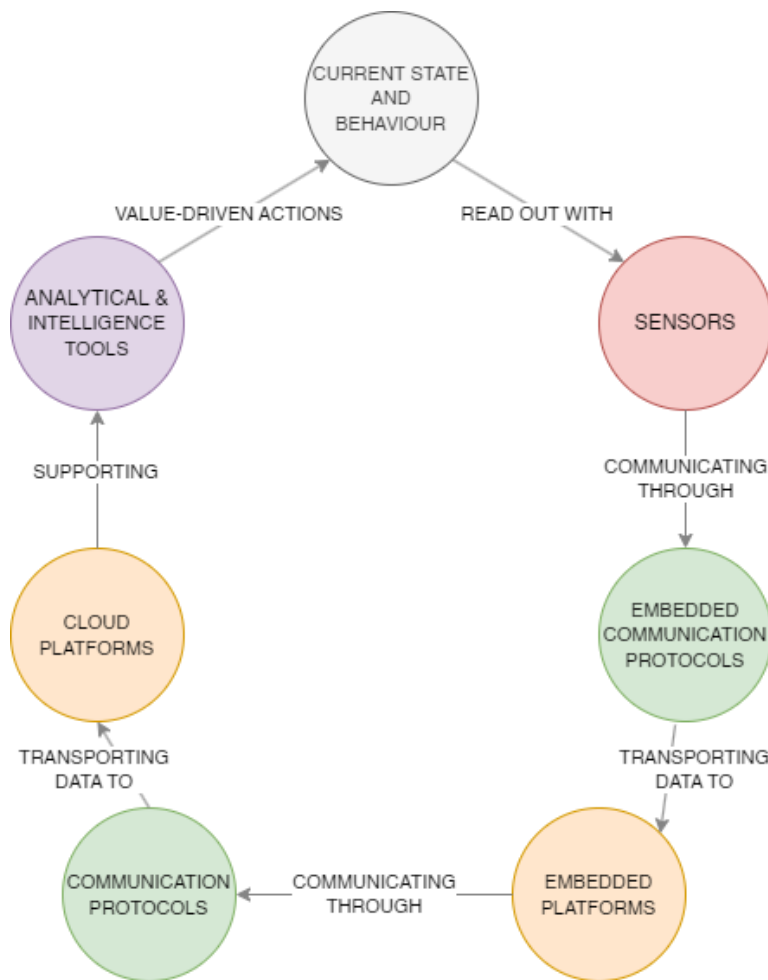
That was our group's motivation to make a system that can (peacefully) neutralize possible attackers, making sure we stay safe. Think of it as a home security system that doesn't only alert you, but also takes some action to scare away intruders.

Now of course we can't just make a machine that harms possible intruders, because you never know if they just come to visit or actually want to cause harm. That is why we try to neutralize them in a fun, but still scary way, rather than going full offence mode.

To put it very simple, our system detects a possible intruder with sensors and camera's using AI, which then moves everything into place making sure to spray the intruder with some harmless water and if they get to close we temporarily blind them with a smoke screen.



## 2.3. Information value loop



### 2.3.1. Current state and behavior

We live in a world where home security becomes more important. More and more homes are getting private security options to scare away burglars. We just think that a simple alarm isn't going to fulfill our needs for long. Because of this we want to try to scare away potentially dangerous people with soft offense. We try to come up with fun and relatively harmless ways to scare away possible intruders and attackers.

### 2.3.2. Read out with sensors

We use a combination of a camera with AI and a distance sensor to detect possible dangers. We have kept possible implementation of many more sensors in mind while designing, allowing for relatively implementation of a light sensor, a sensor for sound, temperature, ... anything you think would make the device even safer

### 2.3.3. Communicate through embedded communication protocols

We use SPI in order to display some of our immediate results on an LCD screen mounted on the side. Extra sensors for the temperature will use the I<sup>2</sup>C protocol in order to communicate their results.

### 2.3.4. Transport data to Embedded platforms

We use a strong combination of both the Raspberry PI pico and the OrangePI 3 LTS to process our data and create a possible reaction. They form our global command center. Here the AI operates depending on the camera input, the distance is measured, other sensor data is collected, the movement is controlled... everything required to guarantee safety.

### 2.3.5. Communicating through communication protocols

These two embedded device can only be strong if they have a stable connection to share data and work together. For this we use an MQTT broker on the Raspberry PI pico which then communicates with the OrangePI 3 LTS.

### 2.3.6. Transporting data to cloud platforms

All of our data could be made available on online cloud platforms. Using ThingSpeak for sensor data and uBEAC to combine all data together and eventually visualize it.

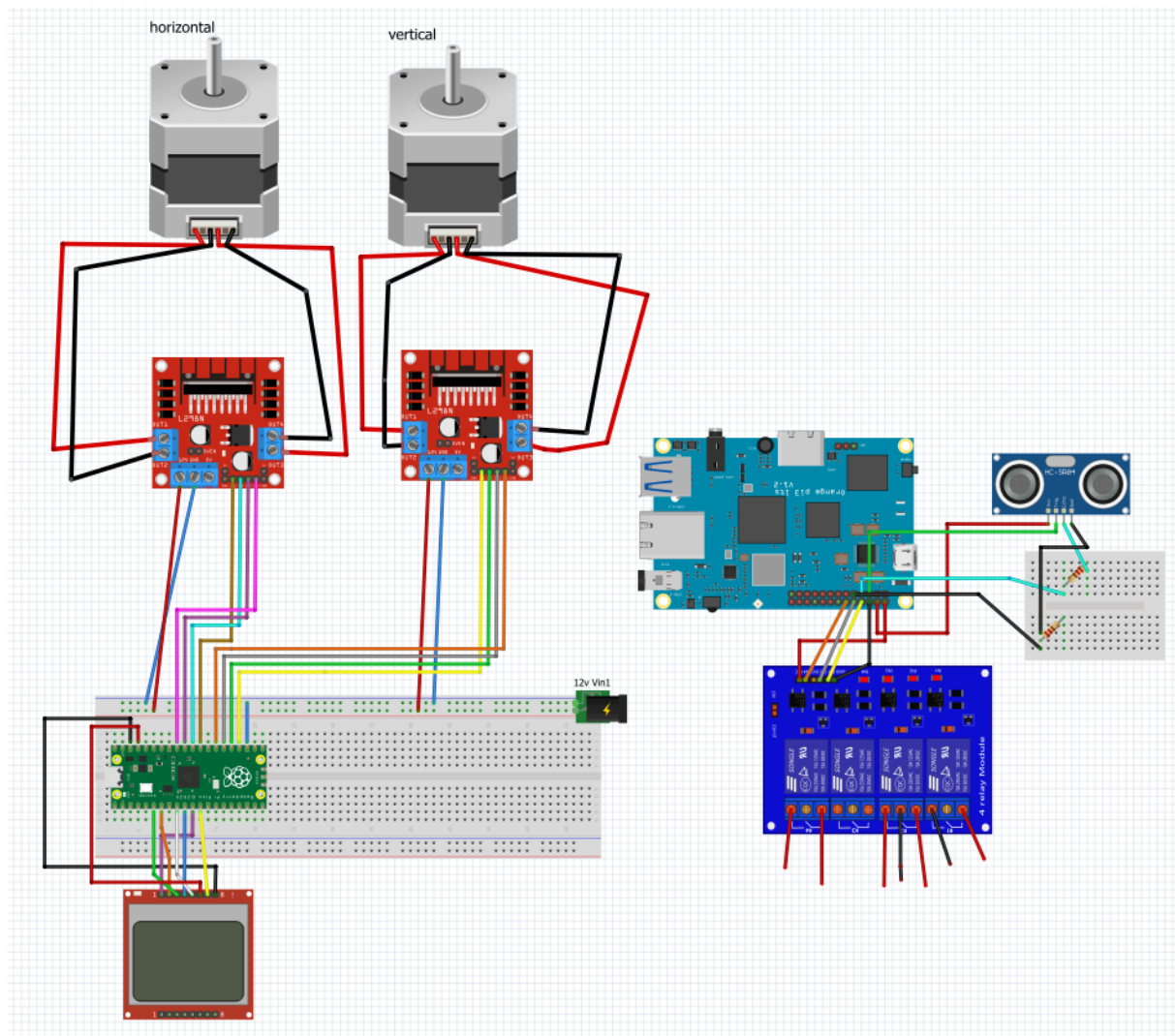
### 2.3.7. Supporting Analytical and intelligence tools

As mentioned before, our system uses a camera with AI. This AI detects where a person is and visualizes it for the user to see. The motors operate on commands given by the AI, communicated over MQTT.

### 2.3.8. Value driven actions

By this way we believe that our system can defend your home in a better way than the average alarm system, while not being too harmful. The system will align itself carefully with the opponent and surprise attack it with a blast of water they will never see coming, while being blinded by a smokescreen.

## 2.4. Schematic



## 2.5. Pictures







## 3. Project code

### 3.1. Code printout

#### 3.1.1. Raspberry PI Pico

```
import board
import busio
import digitalio
import adafruit_pcd8544
import wifi
import socketpool
import adafruit_minimqtt.adafruit_minimqtt as MQTT
import json
import time

# WiFi connection details
wifi_ssid = '*****' # Sensored because someone put their actual data
in here :D
wifi_password = '*****' # Sensored because someone put their actual
data in here :D

# Connect to WiFi
print("Connecting to WiFi...")
try:
    wifi.radio.connect(wifi_ssid, wifi_password)
    print("Connected to WiFi!")
except Exception as e:
    print(f"Failed to connect to WiFi: {e}")
    import sys
    sys.exit()

# MQTT broker details
mqtt_broker = "192.168.0.125"
mqtt_topic = "/home/data"

# MQTT setup
pool = socketpool.SocketPool(wifi.radio)
mqtt_client = MQTT.MQTT(
    broker=mqtt_broker,
    port=1883,
    socket_pool=pool,
)

def connect(client, userdata, flags, rc):
```

```

    print("Connected to MQTT Broker!")
    client.subscribe(mqtt_topic)

def disconnect(client, userdata, rc):
    print("Disconnected from MQTT Broker!")

# Setup for LCD
spi = busio.SPI(clock=board.GP6, MOSI=board.GP7)
cs = digitalio.DigitalInOut(board.GP5)
dc = digitalio.DigitalInOut(board.GP4)
rst = digitalio.DigitalInOut(board.GP8)
back_light = digitalio.DigitalInOut(board.GP9)

cs.direction = digitalio.Direction.OUTPUT
dc.direction = digitalio.Direction.OUTPUT
rst.direction = digitalio.Direction.OUTPUT
back_light.direction = digitalio.Direction.OUTPUT

lcd = adafruit_pcd8544.PCD8544(spi, dc, cs, rst)
lcd.contrast = 50
lcd.rotation = 2

def display_direction(direction):
    lcd.fill(0)
    lcd.text("Direction:", 0, 0, 1)
    lcd.text(direction, 0, 10, 1)
    lcd.show()
    back_light.value = True

def display_center(direction):
    lcd.fill(0)
    lcd.text("Person in center:", 0, 0, 1)
    lcd.text(direction, 0, 10, 1)
    lcd.show()
    back_light.value = True

# Stepper motor control setup
FULL_STEP_SEQUENCE = [
    [1, 1, 0, 0], # Step 1
    [0, 1, 1, 0], # Step 2
    [0, 0, 1, 1], # Step 3
    [1, 0, 0, 1]
]

DELAY_SECONDS = 0.001

```



```

motor1_pins = [
    digitalio.DigitalInOut(board.GP18),
    digitalio.DigitalInOut(board.GP19),
    digitalio.DigitalInOut(board.GP20),
    digitalio.DigitalInOut(board.GP21)
]

motor2_pins = [
    digitalio.DigitalInOut(board.GP10),
    digitalio.DigitalInOut(board.GP11),
    digitalio.DigitalInOut(board.GP12),
    digitalio.DigitalInOut(board.GP13)
]

for pin in motor1_pins + motor2_pins:
    pin.direction = digitalio.Direction.OUTPUT

def rotate_full_step_left():
    display_direction("Left")
    for step_sequence in FULL_STEP_SEQUENCE:
        motor1_pins[0].value = step_sequence[0]
        motor1_pins[1].value = step_sequence[1]
        motor1_pins[2].value = step_sequence[2]
        motor1_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def rotate_full_step_right():
    display_direction("Right")
    for step_sequence in reversed(FULL_STEP_SEQUENCE):
        motor1_pins[0].value = step_sequence[0]
        motor1_pins[1].value = step_sequence[1]
        motor1_pins[2].value = step_sequence[2]
        motor1_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def rotate_full_step_up():
    display_direction("Up")
    for step_sequence in FULL_STEP_SEQUENCE:
        motor2_pins[0].value = step_sequence[0]
        motor2_pins[1].value = step_sequence[1]
        motor2_pins[2].value = step_sequence[2]
        motor2_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def rotate_full_step_down():
    display_direction("Down")
    for step_sequence in reversed(FULL_STEP_SEQUENCE):

```

```

        motor2_pins[0].value = step_sequence[0]
        motor2_pins[1].value = step_sequence[1]
        motor2_pins[2].value = step_sequence[2]
        motor2_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def message(client, topic, message):
    print(f"Received message on topic {topic}: {message}")
    data = json.loads(message)

    left_shoulder = data.get("left_shoulder")
    right_shoulder = data.get("right_shoulder")
    left_hip = data.get("left_hip")
    right_hip = data.get("right_hip")
    margin = data.get("margin")
    center_x = data.get("center_x")
    center_y = data.get("center_y")

    if None in [left_shoulder, right_shoulder, left_hip, right_hip, margin,
center_x, center_y]:
        print('Invalid data received')
        return

    shoulder_hip_points = [left_shoulder, right_shoulder, left_hip,
right_hip]

    if shoulder_hip_points[1][0] > center_x + margin and
shoulder_hip_points[1][1] > center_y + margin:
        print('bottom right diagonal')
        rotate_full_step_right()
        rotate_full_step_up()
    elif shoulder_hip_points[0][0] < center_x - margin and
shoulder_hip_points[0][1] > center_y + margin:
        print('bottom left diagonal')
        rotate_full_step_left()
        rotate_full_step_up()
    elif shoulder_hip_points[3][0] > center_x + margin and
shoulder_hip_points[3][1] < center_y - margin:
        print('top right diagonal')
        rotate_full_step_down()
        rotate_full_step_right()
    elif shoulder_hip_points[2][0] < center_x - margin and
shoulder_hip_points[2][1] < center_y - margin:
        print('top left diagonal')
        rotate_full_step_down()
        rotate_full_step_left()

```

```

    elif shoulder_hip_points[1][0] > center_x - margin and
shoulder_hip_points[3][0] > center_x - margin:
        print('right')
        rotate_full_step_right()
    elif shoulder_hip_points[2][1] < center_y + margin and
shoulder_hip_points[3][1] < center_y + margin:
        print('above')
        rotate_full_step_up()
    elif shoulder_hip_points[0][1] > center_y - margin and
shoulder_hip_points[1][1] > center_y - margin:
        print('beneath')
        rotate_full_step_down()
    elif shoulder_hip_points[0][0] < center_x + margin and
shoulder_hip_points[2][0] < center_x + margin:
        print('left')
        rotate_full_step_left()
    elif shoulder_hip_points[0][0] > center_x - margin and
shoulder_hip_points[0][1] < center_y - margin and \
        shoulder_hip_points[1][0] < center_x + margin and
shoulder_hip_points[1][1] < center_y - margin and \
        shoulder_hip_points[2][0] > center_x - margin and
shoulder_hip_points[2][1] > center_y + margin and \
        shoulder_hip_points[3][0] < center_x + margin and
shoulder_hip_points[3][1] > center_y - margin:
        print('center')
        display_center("shooting")

mqtt_client.on_connect = connect
mqtt_client.on_disconnect = disconnect
mqtt_client.on_message = message

# Try connecting to the MQTT broker with debug information
print("Attempting to connect to MQTT broker...")
try:
    mqtt_client.connect()
    print("Connected to MQTT broker!")
except Exception as e:
    print("Failed to connect to MQTT broker:", e)
    import sys
    sys.exit()

# Keep the program running to receive messages
try:
    while True:
        mqtt_client.loop(timeout=2)
        time.sleep(0.1)

```

```
finally:
    # Clean up GPIO pins
    for pin in motor1_pins + motor2_pins:
        pin.deinit()
    lcd.fill(0)
    lcd.show()
```

### 3.1.2. OrangePI 3 LTS

#### 3.1.2.1. Main

*This code part lacks a few comments, more information on the separate topics can be found in 3.2.2*

```
import cv2
import mediapipe as mp
import numpy as np
import wiringpi as wp
import time
import paho.mqtt.client as mqtt
import json # Import the json module
from NS_hcsr04 import HCSR04

mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5,
min_tracking_confidence=0.5)

broker_address = "192.168.0.125"
topic = "/home/data"

# Define GPIO pins connected to the L298N motor driver
TRIGGER_PIN = 3 # Trigger pin for HC-SR04
ECHO_PIN = 4 # Echo pin for HC-SR04

RELAY1_PIN = 1 # Relay 1 connected to GPIO pin 1
RELAY2_PIN = 2
RELAY4_pin=5

# Function to initialize GPIO pins and WiringPi library
wp.wiringPiSetup()
wp.pinMode(RELAY1_PIN, 1)
wp.pinMode(RELAY2_PIN, 1)
wp.pinMode(RELAY4_pin,1)

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker!")
    else:
        print("Failed to connect, return code %d\n", rc)
```

```

def detect_shoulder_hip(image):
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image_rgb)

    if results.pose_landmarks:
        landmarks = results.pose_landmarks.landmark
        left_shoulder =
(int(landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x *
image.shape[1]),
                    int(landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.va
lue].y * image.shape[0]))
        right_shoulder =
(int(landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x *
image.shape[1]),
                    int(landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.
value].y * image.shape[0]))
        left_hip = (int(landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x *
image.shape[1]),
                    int(landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y *
image.shape[0]))
        right_hip = (int(landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x *
image.shape[1]),
                    int(landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y *
image.shape[0]))

        return left_shoulder, right_shoulder, left_hip, right_hip
    else:
        return None

def enableRelay(pin):
    wp.digitalWrite(pin, 1)
    print(f"Relay on pin {pin} enabled")

def disableRelay(pin):
    wp.digitalWrite(pin, 0)
    print(f"Relay on pin {pin} disabled")
def trapezoid_area(left_top, left_bottom, right_top, right_bottom):
    width_top = np.linalg.norm(np.array(right_top) - np.array(left_top))
    width_bottom = np.linalg.norm(np.array(right_bottom) -
np.array(left_bottom))
    height = np.linalg.norm(np.array(left_bottom) - np.array(left_top))
    area = 0.5 * (width_top + width_bottom) * height
    return area

```

```

def calculate_margin(area):
    min_margin = 10
    max_margin = 50
    margin = min_margin + (max_margin - min_margin) * (area / 100000)
    return int(max(min_margin, min(max_margin, margin)))

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION1)
client.on_connect = on_connect
client.connect(broker_address)
client.loop_start()

cap = cv2.VideoCapture(1, cv2.CAP_V4L2)

while True:
    ret, frame = cap.read()
    sensor = HCSR04(trigger_pin=12, echo_pin=14)

    distance = sensor.distance_cm()

    if distance == 'Out of range':
        distance = 10000

    print(distance)

    if not ret:
        break

    shoulder_hip_points = detect_shoulder_hip(frame)

    if shoulder_hip_points:
        left_shoulder, right_shoulder, left_hip, right_hip =
        shoulder_hip_points
        if shoulder_hip_points is not None:
            area = trapezoid_area(left_shoulder, left_hip, right_shoulder,
            right_hip)
            margin = calculate_margin(area)
            center_x = frame.shape[1] // 2
            center_y = frame.shape[0] // 2

            # Convert the points to a dictionary
            data_dict = {
                "left_shoulder": left_shoulder,
                "right_shoulder": right_shoulder,
                "left_hip": left_hip,

```

```

        "right_hip": right_hip,
        "margin":margin,
        "center_y": center_y,
        "center_x":center_x
    }

    # Convert the dictionary to a JSON string
    data_json = json.dumps(data_dict)

    # Publish the JSON string
    client.publish(topic, data_json)

    print("Message sent: ", data_json)

    for point in shoulder_hip_points:
        cv2.circle(frame, point, 5, (0, 255, 0), -1)

        cv2.line(frame, shoulder_hip_points[0], shoulder_hip_points[1],
(255, 255, 255), 2)
        cv2.line(frame, shoulder_hip_points[2], shoulder_hip_points[3],
(255, 255, 255), 2)
        cv2.line(frame, shoulder_hip_points[0], shoulder_hip_points[2],
(255, 255, 255), 2)
        cv2.line(frame, shoulder_hip_points[1], shoulder_hip_points[3],
(255, 255, 255), 2)

    if distance >= 100 or distance == 0 :
        enableRelay(RELAY2_PIN)
        enableRelay(RELAY1_PIN)
        time.sleep(1) # Ensure RELAY2_PIN is enabled for at least 1 second
        disableRelay(RELAY2_PIN)
    else:
        disableRelay(RELAY1_PIN)
        disableRelay(RELAY2_PIN)

    if shoulder_hip_points is None:
        print('empty')
    else:
        if shoulder_hip_points[0][0] > center_x - margin and
shoulder_hip_points[0][1] < center_y - margin and \
            shoulder_hip_points[1][0] < center_x + margin and
shoulder_hip_points[1][1] < center_y - margin and \
            shoulder_hip_points[2][0] > center_x - margin and
shoulder_hip_points[2][1] > center_y + margin and \
            shoulder_hip_points[3][0] < center_x + margin and
shoulder_hip_points[3][1] > center_y - margin:

```



```

        print('center')
        wp.digitalWrite(RELAY4_pin,1)
    else:
        wp.digitalWrite(RELAY4_pin,0)

    #cv2.circle(frame, (center_x, center_y), 5, (0, 0, 255), -1)
    # cv2.imshow("Shoulder and Hip Points", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

```

### 3.1.2.2. Library NS\_HCSR04.py

```

import wiringpi as wp
import time

__version__ = '0.6.9'
__author__ = 'NanoSievert'
__license__ = "BSD 3-Clause"

class HCSR04:
    """
    Unofficial port of HC-SR04 Driver by rsc1975 made for an IOT project
    trying to blast people with a pink water gun (makes it support wiringpi)
    """
    # Util functions
    def convert_us_s(us):
        """Converts time in microseconds to seconds."""
        return us*(10**(-6))

    # Main functions

    def __init__(self, trigger_pin, echo_pin, echo_timeout_us=500*2*30):
        """
        trigger_pin: Output pin to send pulses
        echo_pin: Readonly pin to measure the distance. The pin should be
        protected with 1k resistor
        echo_timeout_us: Timeout in microseconds to listen to echo pin.
        By default is based in sensor limit range (4m)
        """
        self.echo_timeout_us = echo_timeout_us

```

```

self.trigger_pin = trigger_pin
self.echo_pin = echo_pin

# Setup WiringPi
wp.wiringPiSetup()

# Set trigger pin as output
wp.pinMode(self.trigger_pin, 1)

# Set echo pin as input
wp.pinMode(self.echo_pin, 0)

def _send_pulse_and_wait(self):
    """
    Send the pulse to trigger and listen on echo pin.
    We use the method `wp.micros()` to get the microseconds until the
    echo is received.
    """
    wp.digitalWrite(self.trigger_pin, wp.LOW)
    time.sleep(0.000005) # Stabilize the sensor
    wp.digitalWrite(self.trigger_pin, wp.HIGH)
    time.sleep(0.00001) # Send a 10us pulse
    wp.digitalWrite(self.trigger_pin, wp.LOW)

    # Start time
    start = time.time()
    # Wait for the echo pin to go high
    while wp.digitalRead(self.echo_pin) == wp.LOW:
        if (time.time() - start) > (self.echo_timeout_us / 1000000):
            print('Out of range')
            return

    start = time.time()
    # Wait for the echo pin to go low
    while wp.digitalRead(self.echo_pin) == wp.HIGH:
        pulse_time = (time.time() - start) * 1000000 # Convert to
        microseconds
        if pulse_time > self.echo_timeout_us:
            print('Out of range')
            return

    return pulse_time

def distance_mm(self):
    """
    Get the distance in millimeters without floating point operations.
    """

```

```

pulse_time = self._send_pulse_and_wait()

# To calculate the distance we get the pulse_time and divide it by 2
# (the pulse walk the distance twice) and by 29.1 because
# the sound speed on air (343.2 m/s), that It's equivalent to
# 0.34320 mm/us that is 1mm each 2.91us
# pulse_time // 2 // 2.91 -> pulse_time // 5.82 -> pulse_time * 100
// 582

if not isinstance(pulse_time, float):
    return 0

mm = pulse_time * 100 // 582
return mm

def distance_cm(self):
    """
    Get the distance in centimeters with floating point operations.
    It returns a float.
    """
    pulse_time = self._send_pulse_and_wait()

    # To calculate the distance we get the pulse_time and divide it by 2
    # (the pulse walk the distance twice) and by 29.1 because
    # the sound speed on air (343.2 m/s), that It's equivalent to
    # 0.034320 cm/us that is 1cm each 29.1us

    if not isinstance(pulse_time, float):
        return 0

    cms = (pulse_time / 2) / 29.1
    return cms

```

## 3.2. Detailed Code explanation

### 3.2.1. Raspberry PI Pico

#### 3.2.1.1. Main file

##### IMPORTS

```
import board
import busio
import digitalio
import adafruit_pcd8544
import wifi
import socketpool
import adafruit_minimqtt.adafruit_minimqtt as MQTT
import json
import time
```

*Importing all required libraries*

##### CONNECT TO WIFI

```
# WiFi connection details
wifi_ssid = '*****' # Sensored because someone put their actual data
in here :D
wifi_password = '*****' # Sensored because someone put their actual
data in here :D

# Connect to WiFi
print("Connecting to WiFi...")
try:
    wifi.radio.connect(wifi_ssid, wifi_password)
    print("Connected to WiFi!")
except Exception as e:
    print(f"Failed to connect to WiFi: {e}")
    import sys
    sys.exit()
```

- Wi-Fi ssid and password defined in variables
- Wifi module used to connect to wifi (exception statement)
  - o If success: => print message
  - o If fails: => exit program

## SETUP MQTT BROKER

```
# MQTT broker details
mqtt_broker = "192.168.0.125"
mqtt_topic = "/home/data"

# MQTT setup
pool = socketpool.SocketPool(wifi.radio)
mqtt_client = MQTT.MQTT(
    broker=mqtt_broker,
    port=1883,
    socket_pool=pool,
)

def connect(client, userdata, flags, rc):
    print("Connected to MQTT Broker!")
    client.subscribe(mqtt_topic)

def disconnect(client, userdata, rc):
    print("Disconnected from MQTT Broker!")
```

- Details stored in variables
- Library socketpool used to use the wifi as local socket pool
- MQTT library used to store mqtt client in variable
- Connect function used to subscribe to a certain MQTT topic

## LCD

```

# Setup for LCD
spi = busio.SPI(clock=board.GP6, MOSI=board.GP7)
cs = digitalio.DigitalInOut(board.GP5)
dc = digitalio.DigitalInOut(board.GP4)
rst = digitalio.DigitalInOut(board.GP8)
back_light = digitalio.DigitalInOut(board.GP9)

cs.direction = digitalio.Direction.OUTPUT
dc.direction = digitalio.Direction.OUTPUT
rst.direction = digitalio.Direction.OUTPUT
back_light.direction = digitalio.Direction.OUTPUT

lcd = adafruit_pcd8544.PCD8544(spi, dc, cs, rst)
lcd.contrast = 50
lcd.rotation = 2

def display_direction(direction):
    lcd.fill(0)
    lcd.text("Direction:", 0, 0, 1)
    lcd.text(direction, 0, 10, 1)
    lcd.show()
    back_light.value = True

def display_center(direction):
    lcd.fill(0)
    lcd.text("Person in center:", 0, 0, 1)
    lcd.text(direction, 0, 10, 1)
    lcd.show()
    back_light.value = True

```

- Define variables containing details
  - o Spi => use busio library
  - o Cs/dc/rst/back\_light => setup pins for input/output mode (similar to wiringPI)
- Define directions
- Setup visual preferences for LCD
- Print data on screen
  - o Display direction if not center => also say where
  - o Display when person in center

## STEPPER MOTOR CONTROL

### DEFINE STEP SEQUENCE

```
FULL_STEP_SEQUENCE = [  
    [1, 1, 0, 0], # Step 1  
    [0, 1, 1, 0], # Step 2  
    [0, 0, 1, 1], # Step 3  
    [1, 0, 0, 1]  
]
```

Define which parts should be powered on

### SETUP PREFERENCES AND PINS USING DIGITALIO LIBRARY

```
DELAY_SECONDS = 0.001  
  
motor1_pins = [  
    digitalio.DigitalInOut(board.GP18),  
    digitalio.DigitalInOut(board.GP19),  
    digitalio.DigitalInOut(board.GP20),  
    digitalio.DigitalInOut(board.GP21)  
]  
  
motor2_pins = [  
    digitalio.DigitalInOut(board.GP10),  
    digitalio.DigitalInOut(board.GP11),  
    digitalio.DigitalInOut(board.GP12),  
    digitalio.DigitalInOut(board.GP13)  
]  
  
for pin in motor1_pins + motor2_pins:  
    pin.direction = digitalio.Direction.OUTPUT
```

- Delay set to 1 ms
- Motor pins defined for both vertical and horizontal motor (GP referencing pins as seen in hardware)
- Setup direction for pins

## DEFINE FUNCTIONS FOR MOVING MOTOR

```
def rotate_full_step_left():
    display_direction("Left")
    for step_sequence in FULL_STEP_SEQUENCE:
        motor1_pins[0].value = step_sequence[0]
        motor1_pins[1].value = step_sequence[1]
        motor1_pins[2].value = step_sequence[2]
        motor1_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def rotate_full_step_right():
    display_direction("Right")
    for step_sequence in reversed(FULL_STEP_SEQUENCE):
        motor1_pins[0].value = step_sequence[0]
        motor1_pins[1].value = step_sequence[1]
        motor1_pins[2].value = step_sequence[2]
        motor1_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def rotate_full_step_up():
    display_direction("Up")
    for step_sequence in FULL_STEP_SEQUENCE:
        motor2_pins[0].value = step_sequence[0]
        motor2_pins[1].value = step_sequence[1]
        motor2_pins[2].value = step_sequence[2]
        motor2_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)

def rotate_full_step_down():
    display_direction("Down")
    for step_sequence in reversed(FULL_STEP_SEQUENCE):
        motor2_pins[0].value = step_sequence[0]
        motor2_pins[1].value = step_sequence[1]
        motor2_pins[2].value = step_sequence[2]
        motor2_pins[3].value = step_sequence[3]
        time.sleep(DELAY_SECONDS)
```



## RECEIVE DATA OVER MQTT + REACT

## RECEIVE MESSAGE OVER MQTT CLIENT

```
def message(client, topic, message):
    print(f"Received message on topic {topic}: {message}")
    data = json.loads(message)

    left_shoulder = data.get("left_shoulder")
    right_shoulder = data.get("right_shoulder")
    left_hip = data.get("left_hip")
    right_hip = data.get("right_hip")
    margin = data.get("margin")
    center_x = data.get("center_x")
    center_y = data.get("center_y")

    if None in [left_shoulder, right_shoulder, left_hip, right_hip, margin,
center_x, center_y]:
        print('Invalid data received')
        return
```

## REACT TO IT BY USING THE STEPPER FUNCTIONS

```
    shoulder_hip_points = [left_shoulder, right_shoulder, left_hip,
right_hip]

    if shoulder_hip_points[1][0] > center_x + margin and
shoulder_hip_points[1][1] > center_y + margin:
        print('bottom right diagonal')
        rotate_full_step_right()
        rotate_full_step_up()
    elif shoulder_hip_points[0][0] < center_x - margin and
shoulder_hip_points[0][1] > center_y + margin:
        print('bottom left diagonal')
        rotate_full_step_left()
        rotate_full_step_up()
    elif shoulder_hip_points[3][0] > center_x + margin and
shoulder_hip_points[3][1] < center_y - margin:
        print('top right diagonal')
        rotate_full_step_down()
        rotate_full_step_right()
```

```

    elif shoulder_hip_points[2][0] < center_x - margin and
shoulder_hip_points[2][1] < center_y - margin:
        print('top left diagonal')
        rotate_full_step_down()
        rotate_full_step_left()
    elif shoulder_hip_points[1][0] > center_x - margin and
shoulder_hip_points[3][0] > center_x - margin:
        print('right')
        rotate_full_step_right()
    elif shoulder_hip_points[2][1] < center_y + margin and
shoulder_hip_points[3][1] < center_y + margin:
        print('above')
        rotate_full_step_up()
    elif shoulder_hip_points[0][1] > center_y - margin and
shoulder_hip_points[1][1] > center_y - margin:
        print('beneath')
        rotate_full_step_down()
    elif shoulder_hip_points[0][0] < center_x + margin and
shoulder_hip_points[2][0] < center_x + margin:
        print('left')
        rotate_full_step_left()
    elif shoulder_hip_points[0][0] > center_x - margin and
shoulder_hip_points[0][1] < center_y - margin and \
        shoulder_hip_points[1][0] < center_x + margin and
shoulder_hip_points[1][1] < center_y - margin and \
        shoulder_hip_points[2][0] > center_x - margin and
shoulder_hip_points[2][1] > center_y + margin and \
        shoulder_hip_points[3][0] < center_x + margin and
shoulder_hip_points[3][1] > center_y - margin:
        print('center')
        display_center("shooting")

```

- Uses earlier defined functions to move the motors depending on the input they get from MQTT server

#### CONNECT AND GET DATA

```

mqtt_client.on_connect = connect
mqtt_client.on_disconnect = disconnect
mqtt_client.on_message = message

```

**CONNECT TO BROKER WITH DEBUG INFORMATION**

```
# Try connecting to the MQTT broker with debug information
print("Attempting to connect to MQTT broker...")
try:
    mqtt_client.connect()
    print("Connected to MQTT broker!")
except Exception as e:
    print("Failed to connect to MQTT broker:", e)
import sys
sys.exit()
```

**KEEP PROGRAM RUNNING TO KEEP RECEIVING DATA**

- Make sure you can keep reacting and moving everything with stepper motors

```
# Keep the program running to receive messages
try:
    while True:
        mqtt_client.loop(timeout=2)
        time.sleep(0.1)
finally:
    # Clean up GPIO pins
    for pin in motor1_pins + motor2_pins:
        pin.deinit()
    lcd.fill(0)
    lcd.show()
```

**3.2.1.2. Libraries**

- Compiled micropython

```

✓  📁 adafruit_minimqtt
   ├── __init__.py
   ├── adafruit_minimqtt.mpy
   ├── matcher.mpy
   ├── adafruit_connection_manager.mpy
   └── adafruit_pcd8544.mpy

```

### 3.2.2. OrangePI 3 LTS

#### 3.2.2.1. Main

##### IMPORTS

```
import cv2
import mediapipe as mp
import numpy as np
import wiringpi as wp
import time
import paho.mqtt.client as mqtt
import json # Import the json module
from NS_hcsr04 import HCSR04
```

*Importing all required libraries*

##### DEFINE VARIABLES

##### AI MODEL

```
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5,
min_tracking_confidence=0.5)
```

##### MQTT BROKER

```
broker_address = "192.168.0.125"
topic = "/home/data"
```

##### PIN DEFINITIONS

##### DISTANCE SENSOR

```
TRIGGER_PIN = 3 # Trigger pin for HC-SR04
ECHO_PIN = 4    # Echo pin for HC-SR04
```

##### RELAYS

```
RELAY1_PIN = 1 # Relay 1 connected to GPIO pin 1
RELAY2_PIN = 2
RELAY4_pin=5
```

```
# Function to initialize GPIO pins and WiringPi library
wp.wiringPiSetup()
wp.pinMode(RELAY1_PIN, 1)
wp.pinMode(RELAY2_PIN, 1)
wp.pinMode(RELAY4_pin,1)
```

#### FUNCTION TO CONNECT TO MQTT BROKER

```
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker!")
    else:
        print("Failed to connect, return code %d\n", rc)
```

## AI FUNCTIONS

## DETECT SHOULDER AND HIP

```
def detect_shoulder_hip(image):
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image_rgb)

    if results.pose_landmarks:
        landmarks = results.pose_landmarks.landmark
        left_shoulder =
(int(landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x *
image.shape[1]),
int(landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.v
alue].y * image.shape[0]))
        right_shoulder =
(int(landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x *
image.shape[1]),
int(landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER
.value].y * image.shape[0]))
        left_hip = (int(landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x *
image.shape[1]),
int(landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y *
image.shape[0]))
        right_hip = (int(landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x
* image.shape[1]),
int(landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y
* image.shape[0]))

        return left_shoulder, right_shoulder, left_hip, right_hip
    else:
        return None
```

- Return positions for every variable

## RELAY FUNCTIONS

```
def enableRelay(pin):
    wp.digitalWrite(pin, 1)
    print(f"Relay on pin {pin} enabled")

def disableRelay(pin):
    wp.digitalWrite(pin, 0)
    print(f"Relay on pin {pin} disabled")
```

#### FUNCTION TO CALCULATE TRAPEZOID AREA

- Used for AI model

```
def trapezoid_area(left_top, left_bottom, right_top, right_bottom):  
    width_top = np.linalg.norm(np.array(right_top) - np.array(left_top))  
    width_bottom = np.linalg.norm(np.array(right_bottom) -  
np.array(left_bottom))  
    height = np.linalg.norm(np.array(left_bottom) - np.array(left_top))  
    area = 0.5 * (width_top + width_bottom) * height  
    return area
```

#### FUNCTION TO CALCULATE MARGIN

- Used for AI model

```
def calculate_margin(area):  
    min_margin = 10  
    max_margin = 50  
    margin = min_margin + (max_margin - min_margin) * (area / 100000)  
    return int(max(min_margin, min(max_margin, margin)))
```

#### CONNECT TO MQTT BROKER

- Uses functions that were defined earlier

```
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION1)  
client.on_connect = on_connect  
client.connect(broker_address)  
client.loop_start()
```

#### INITIALISE OPENCV TO CAPTURE IMAGES FOR AI

```
cap = cv2.VideoCapture(1, cv2.CAP_V4L2)
```

## MAIN PROGRAM LOOP

```
while True:
```

### DEFINE VARIABLES FOR AI

```
ret, frame = cap.read()
```

### DEFINE DISTANCE SENSOR AND GET VARIABLE CONTAINING DISTANCE

```
sensor = HCSR04(trigger_pin=12, echo_pin=14)

distance = sensor.distance_cm()
```

### LOG AND CORRECT DISTANCE VALUE

```
if distance == 'Out of range':
    distance = 10000

print(distance)
```

If value out of range => set to high number, otherwise can't be compared to another number later on

### AI DATA PROCESSING

```
shoulder_hip_points = detect_shoulder_hip(frame)

if shoulder_hip_points:
    left_shoulder, right_shoulder, left_hip, right_hip =
shoulder_hip_points
    if shoulder_hip_points is not None:
        area = trapezoid_area(left_shoulder, left_hip, right_shoulder,
right_hip)
        margin = calculate_margin(area)
        center_x = frame.shape[1] // 2
        center_y = frame.shape[0] // 2
```



```

# Convert the points to a dictionary
data_dict = {
    "left_shoulder": left_shoulder,
    "right_shoulder": right_shoulder,
    "left_hip": left_hip,
    "right_hip": right_hip,
    "margin": margin,
    "center_y": center_y,
    "center_x": center_x
}

# Convert the dictionary to a JSON string
data_json = json.dumps(data_dict)

# Publish the JSON string
client.publish(topic, data_json)

print("Message sent: ", data_json)

for point in shoulder_hip_points:
    cv2.circle(frame, point, 5, (0, 255, 0), -1)

    cv2.line(frame, shoulder_hip_points[0], shoulder_hip_points[1],
(255, 255, 255), 2)
    cv2.line(frame, shoulder_hip_points[2], shoulder_hip_points[3],
(255, 255, 255), 2)
    cv2.line(frame, shoulder_hip_points[0], shoulder_hip_points[2],
(255, 255, 255), 2)
    cv2.line(frame, shoulder_hip_points[1], shoulder_hip_points[3],
(255, 255, 255), 2)

```

- Check if ret valid, otherwise skip to end of loop and restart
- Use predefined function (see earlier) to get points and read them
- Put obtained data in a dictionary
- Serialize to json
- Send data over MQTT so pico can respond
- Visual version only: draw lines on image following body

**CONTROL SMOKE SCREEN**

```

if distance >= 100 or distance == 0 :
    enableRelay(RELAY2_PIN)
    enableRelay(RELAY1_PIN)
    time.sleep(1) # Ensure RELAY2_PIN is enabled for at least 1 second
    disableRelay(RELAY2_PIN)
else:
    disableRelay(RELAY1_PIN)
    disableRelay(RELAY2_PIN)

```

- If distance larger than 100:
  - o Enable relay (opens gate, turning of smoke)
  - o Check if distance == 0 => make sure it does not trigger when module is started
- Else:
  - o Enable smoke screen

**CONTROL WATER GUN**

```

if shoulder_hip_points is None:
    print('empty')
else:
    if shoulder_hip_points[0][0] > center_x - margin and \
    shoulder_hip_points[0][1] < center_y - margin and \
    shoulder_hip_points[1][0] < center_x + margin and \
    shoulder_hip_points[1][1] < center_y - margin and \
    shoulder_hip_points[2][0] > center_x - margin and \
    shoulder_hip_points[2][1] > center_y + margin and \
    shoulder_hip_points[3][0] < center_x + margin and \
    shoulder_hip_points[3][1] > center_y - margin:
        print('center')
        wp.digitalWrite(RELAY4_pin,1)
    else:
        wp.digitalWrite(RELAY4_pin,0)

```

- If screen empty => do nothing
- Else:
  - o Check where person is
  - o If in center => enable relay to spray water

**ALLOW PROCESS TO BE CANCELLED**

```
    if cv2.waitKey(1) & 0xFF == ord('q'):  
        break  
  
cap.release()  
cv2.destroyAllWindows()
```

### 3.2.2.2. Custom module – NS\_hcrs04

- Based on module made by rsc1975 in order for it to support WiringPI

#### IMPORTS

```
import wiringpi as wp
import time
```

#### MODULE INFORMATION

```
__version__ = '0.6.9'
__author__ = 'NanoSievert'
__license__ = "BSD 3-Clause"
```

- Version: don't even worry about it
- Author: NanoSievert = Rik
- License: you can do whatever you want with it (based on original license)
- *Note: most of the documentation is edited from original module*

#### CLASS DEFINITION

- So object can be made of sensor

```
class HCSR04:
    """
    Unofficial port of HC-SR04 Driver by rsc1975 made for an IOT project
    trying to blast people with a pink water gun (makes it support wiringpi)
    """
```

#### UTIL FUNCTIONS

```
# Util functions
def convert_us_s(us):
    """Converts time in microseconds to seconds."""
    return us*(10**(-6))
```

- Converts microseconds ( $\mu$ s) to seconds (s)

## MAIN FUNCTIONS

### CONSTRUCTOR

```
def __init__(self, trigger_pin, echo_pin, echo_timeout_us=500*2*30):  
    """  
    trigger_pin: Output pin to send pulses  
    echo_pin: Readonly pin to measure the distance. The pin should be  
protected with 1k resistor  
    echo_timeout_us: Timeout in microseconds to listen to echo pin.  
By default is based in sensor limit range (4m)  
    """  
  
    self.echo_timeout_us = echo_timeout_us  
    self.trigger_pin = trigger_pin  
    self.echo_pin = echo_pin  
  
    # Setup WiringPi  
    wp.wiringPiSetup()  
  
    # Set trigger pin as output  
    wp.pinMode(self.trigger_pin, 1)  
  
    # Set echo pin as input  
    wp.pinMode(self.echo_pin, 0)
```

- Required parameters:
  - o Trigger\_pin
  - o Echo\_ping
- Optional parameters:
  - o Echo-timeout
- Sets up wiringpi
- Sets trigger pin to output
- Set echo pin to input

**[FUNCTION] SEND PULSE AND WAIT**

```

def _send_pulse_and_wait(self):
    """
    Send the pulse to trigger and listen on echo pin.
    We use the method `wp.micros()` to get the microseconds until the
    echo is received.
    """
    wp.digitalWrite(self.trigger_pin, wp.LOW)
    time.sleep(0.00005) # Stabilize the sensor
    wp.digitalWrite(self.trigger_pin, wp.HIGH)
    time.sleep(0.00001) # Send a 10us pulse
    wp.digitalWrite(self.trigger_pin, wp.LOW)

    # Start time
    start = time.time()
    # Wait for the echo pin to go high
    while wp.digitalRead(self.echo_pin) == wp.LOW:
        if (time.time() - start) > (self.echo_timeout_us / 1000000):
            print('Out of range')
            return

    start = time.time()
    # Wait for the echo pin to go low
    while wp.digitalRead(self.echo_pin) == wp.HIGH:
        pulse_time = (time.time() - start) * 1000000 # Convert to
        microseconds
        if pulse_time > self.echo_timeout_us:
            print('Out of range')
            return

    return pulse_time

```

- Send pulse on trigger pin
- Listen on echo pin
- Measure microseconds between trigger and echo and return that

**[FUNCTION] GET DISTANCE IN MM**

```

def distance_mm(self):
    """
    Get the distance in millimeters without floating point operations.
    """
    pulse_time = self._send_pulse_and_wait()

    # To calculate the distance we get the pulse_time and divide it by
    2
    # (the pulse walk the distance twice) and by 29.1 because
    # the sound speed on air (343.2 m/s), that It's equivalent to
    # 0.34320 mm/us that is 1mm each 2.91us
    # pulse_time // 2 // 2.91 -> pulse_time // 5.82 -> pulse_time * 100
    // 582

    if not isinstance(pulse_time, float):
        return 0

    mm = pulse_time * 100 // 582
    return mm

```

- Use function defined in previous step
- Calculate distance depending on the speed at which sound travels in air
  - o Also keeps in mind that the signal travels both to the object and back from the object, so the first result is divided by two to get the distance
    - *Integrated in the formula*

**[FUNCTION] GET DISTANCE IN CM**

```
def distance_cm(self):  
    """  
    Get the distance in centimeters with floating point operations.  
    It returns a float.  
    """  
    pulse_time = self._send_pulse_and_wait()  
  
    # To calculate the distance we get the pulse_time and divide it by  
    2  
    # (the pulse walk the distance twice) and by 29.1 because  
    # the sound speed on air (343.2 m/s), that It's equivalent to  
    # 0.034320 cm/us that is 1cm each 29.1us  
  
    if not isinstance(pulse_time, float):  
        return 0  
  
    cms = (pulse_time / 2) / 29.1  
    return cms
```

- Similar to previous one, just uses cm as output instead of mm (decided early in calculation process)








## 4. Hardware Explanation


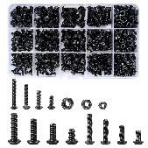

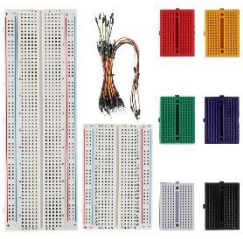



### 4.1. List of components

#### 4.1.1. Standard components

- Jumper cables
- (1x) Raspberry PI pico
- (1x) OrangePI 3 LTS
- (1x) HC-SR04 Sensor (Distance)
- (2x) 220 ohm resistor
- (1x) LCD 5110

#### 4.1.2. Extra components

Object(s)	Image
(1x) Watergun	
(5x) L298N Motor Driver	
(1x) Smoke machine	
(2x) 12V Battery holders	
(5x) Nema 17	

<b>(4x) Shaft coupler</b>	
<b>Nuts and bolts</b>	
<b>PLA+ filament</b>	
<b>Smaller breadboards</b>	
<b>Adhesive pads</b>	
<b>Furniture feet</b>	
<b>Ball bearings</b>	

## 4.2. Components

### 4.2.1. Jumper cables



Source	IOT-ESSENTIALS-Kit
Functionalities	<ul style="list-style-type: none"> <li>- Connect things</li> <li>- Deliver power</li> </ul>

### 4.2.2. (1x) Raspberry PI pico

#### 4.2.2.1. The microcontroller



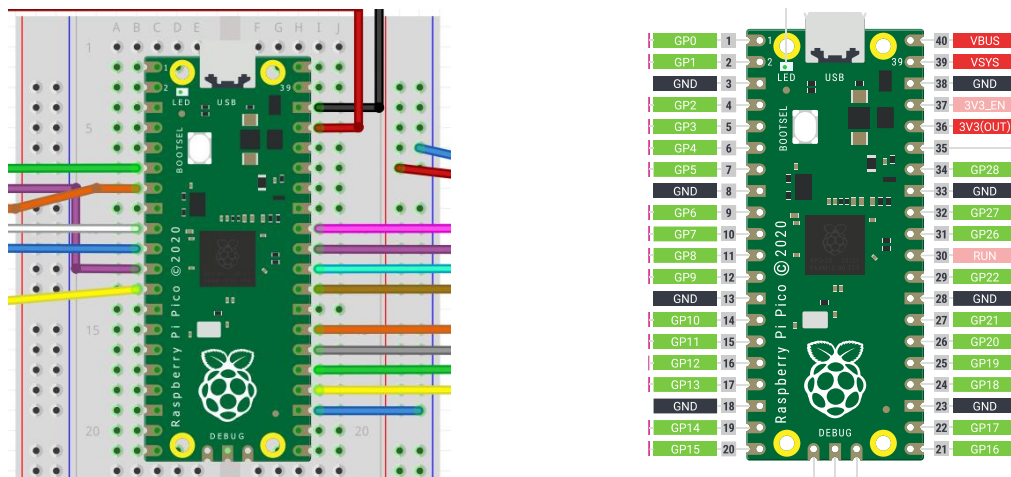
Feature	Description
Processor	Dual-core Arm Cortex M0+ processor
Clock Speed	Flexible clock running up to 133 MHz
SRAM	264kB
Flash Memory	2MB on-board
USB	USB 1.1 with device and host support
Power Modes	Low-power sleep and dormant modes
Programming	Drag-and-drop programming using mass storage over USB
GPIO Pins	26 x multi-function GPIO pins
SPI	2 x SPI
I2C	2 x I2C
UART	2 x UART
ADC	3 x 12-bit ADC
PWM Channels	16 x controllable PWM channels

<b>Clock and Timer</b>	Accurate clock and timer on-chip
<b>Temperature Sensor</b>	Yes
<b>Floating-Point Libraries</b>	Accelerated floating-point libraries on-chip
<b>Programmable I/O (PIO) State Machines</b>	8 × Programmable I/O (PIO) state machines for custom peripheral support

#### 4.2.2.2. Project Integration

Source	Given in class
<b>Functionalities</b>	<ul style="list-style-type: none"> <li>- Receive messages</li> <li>- Turn motors</li> <li>- Display things on LCD</li> </ul>

In our project, this component has been mounted on a breadboard to make its pins more accessible.

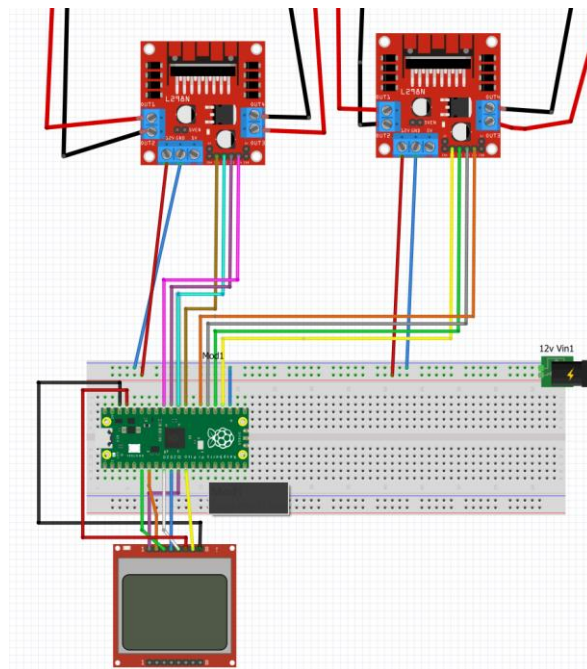


#### 4.2.2.3. Connections (pin-based)

G°	Color schematic	Usage	Source	Destination
6	Lime [L]	SPIO	[PICO] SPI0 RX	[LCD] DC
7	Orange [L]	SPIO	[PICO] SPI0 CSn	[LCD] CD
9	White [L]	GP	[PICO] SPI0 SCK	[LCD] CLK
10	Blue [L]	GP	[PICO] SPI0 TX	[LCD] DIN
11	Purple [L]	GP	[PICO] SPI1 RX	[LCD] RST
12	Yellow [L]	GP	[PICO] SPI1 CSn	[LCD] BL
23	Blue [R]	GND	[PICO] GND	[L298N] (V+H) GND
24	Yellow [R]	GP	[PICO] GP-18	[L298N] (V) IN 1
25	Lime [R]	GP	[PICO] GP-19	[L298N] (V) IN 2
26	Gray [R]	GP	[PICO] GP-20	[L298N] (V) IN 3

27	Orange [R]	GP	[PICO] GP-21	[L298N] (V) IN 4
29	Brown [R]	GP	[PICO] GP-22	[L298N] (H) IN 1
30	Aqua [R]	GP	[PICO] RUN	[L298N] (H) IN 2
31	Purple [R]	GP	[PICO] GP-26	[L298N] (H) IN 3
32	Pink [R]	GP	[PICO] GP-27	[L298N] (H) IN 4
37	Red [R]	3V3	[PICO] 3.3V out	[LCD] VCC
38	Black [R]	GND	[PICO] GND	[LCD] GND

- Connections used to control LCD: 6, 7, 9, 10, 11, 12, (Power: 37, 38)
- Connections used to control motors:
  - o Vertical: 24, 25, 26, 27
  - o Horizontal: 29, 30, 31, 32
  - o Power provided through breadboard (12V external input)



### 4.2.3. (1x) OrangePI 3 LTS

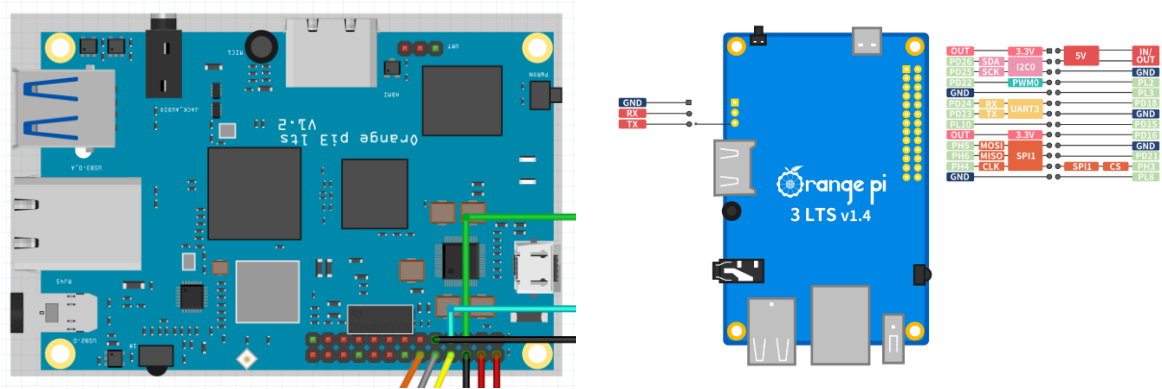
#### 4.2.3.1. The embeded device



Feature	Description
Processor	Allwinner H6 Quad-core Cortex-A53
GPU	Mali T720MP2 GPU
Memory	2GB DDR3 (shared with GPU)
Storage Options	microSD card slot, 8GB eMMC flash
Networking	Gigabit Ethernet, Wi-Fi (802.11 b/g/n/ac), Bluetooth 5.0
USB Ports	3 × USB 2.0, 1 × USB 3.0 OTG
Display Outputs	HDMI 2.0a (up to 4K), CVBS, DSI
Camera Interface	CSI Camera Connector
Audio	HDMI output, 3.5mm audio jack
Expansion	26-pin GPIO header
Power Supply	5V/3A via USB Type-C
Operating System Support	Android, Debian, Ubuntu
Dimensions	90mm × 64mm
Additional Features	IR receiver, RTC battery connector

#### 4.2.3.2. Project Integration

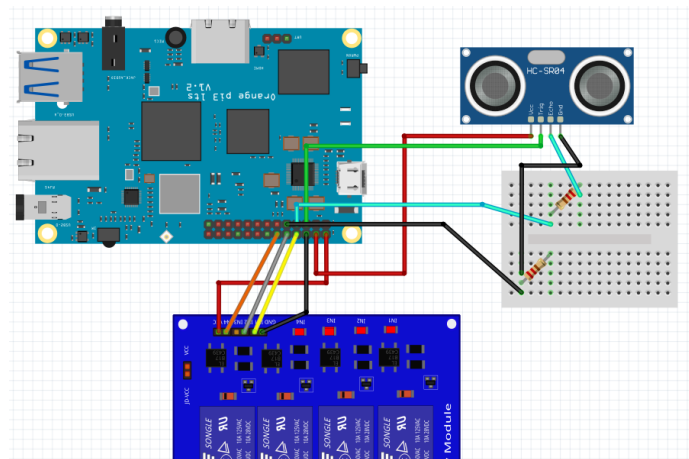
Source	Given in class
Functionalities	<ul style="list-style-type: none"> <li>- Send messages</li> <li>- Runs AI Model</li> <li>- Triggers relays               <ul style="list-style-type: none"> <li>o Gun</li> <li>o Smoke machine</li> </ul> </li> <li>- Operate distance sensor</li> </ul>



#### 4.2.3.3. Connections (pin-based)

G°	Color schematic	Usage	Source	Destination
2	Red [R]	5V out	[OPI3] 5V out	[RELAY] Vcc
4	Red [R]	5V out	[OPI3] 5V out	[HC] Vcc
5	Lime [L]	GPIO-1	[OPI3] GPIO-1	[HC] Trigger
6	Black [R]	GND	[OPI3] GND	[RELAY] GND
7	Aqua [L]	GPIO-2	[OPI3] GPIO-2	[HC] Echo
8	Yellow [R]	GPIO-3	[OPI3] GPIO-3	[RELAY] IN 1
9	Black [L]	GND	[OPI3] GND	[HC] GND
10	Gray [R]	GPIO-4	[OPI3] GPIO-4	[RELAY] IN 2
12	Orange [R]	GPIO-6	[OPI3] GPIO-6	[RELAY] IN 4

- Connections used to control relay:
  - o Power supply: 2
  - o Ground: 6
  - o Trigger: 8, 10, 12
- Distance Sensor
  - o Power supply: 4
  - o Ground: 9
  - o Trigger: 5
  - o Echo: 7



#### 4.2.4. (1x) HC-SR04 Sensor (Distance)

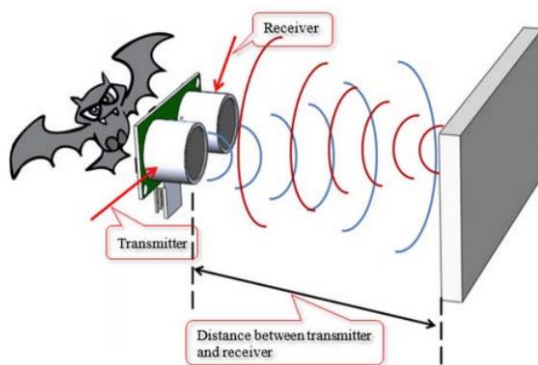
##### 4.2.4.1. The sensor



<b>Component</b>	HC-SR04 Ultrasonic Sensor Module
<b>Manufacturer</b>	Handson Technology

	Datasheet / User Manual	<a href="https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf">https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf</a>
---	-------------------------	---

##### 4.2.4.2. How does it work?



**Figure-1**

1. Module sends ultrasonic signal
2. Signal travels to the object
3. Signal bounces back from object to module
4. Module receives signal
5. Distance is calculated depending on the time it took the signal to go from the module to the object and back

⇒ The output should be divided by 2 because it contains both the way there and the way back!

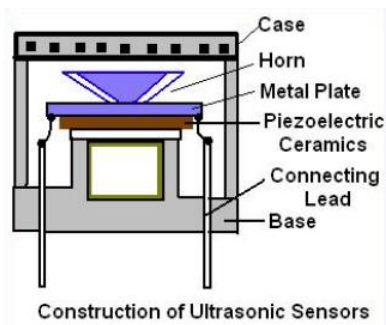


#### 4.2.4.3. Module specifications

Source: Datasheet

Electrical Parameter	Value
Operating Voltage	3.3Vdc ~ 5Vdc
Quiescent Current	<2mA
Operating Current	15mA
Operating Frequency	40KHz
Operating Range & Accuracy	2cm ~ 400cm (1in ~ 13ft) $\pm$ 3mm
Sensitivity	-65dB min
Sound Pressure	112dB
Effective Angle	15°
Connector	4-pins header with 2.54mm pitch
Dimension	45mm x 20mm x 15mm
Weight	9g

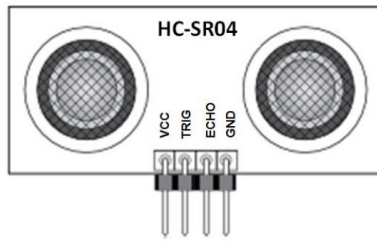
#### 4.2.4.4. Sensor element construction



*Piezoelectric crystals:*

- Oscillate at high frequencies when electricity is applied to it
- Generate electricity when high frequencies hit them.

#### 4.2.4.5. Hardware information



VCC	3V3 – 5V (Note: use resistor when using 5V, otherwise we get magic smoke D: )
TRIG	Triggering Input pin
ECHO	TTL Logic output pin
GND	Ground pin

#### 4.2.4.6. Timing

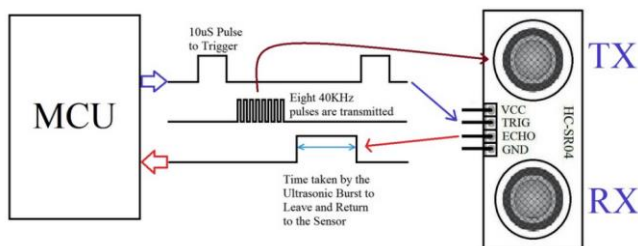
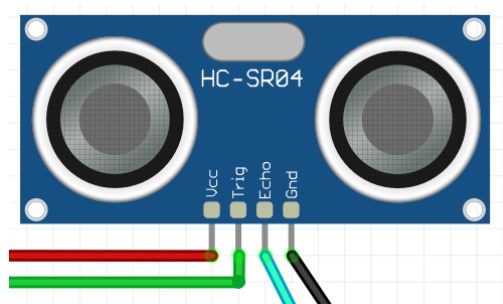
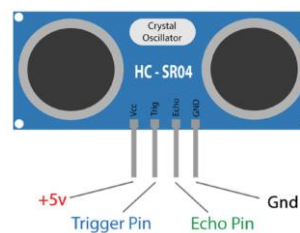


Figure-5: Microcontroller Interfacing

#### 4.2.4.7. Project integration



#### HC-SR04 Pinout

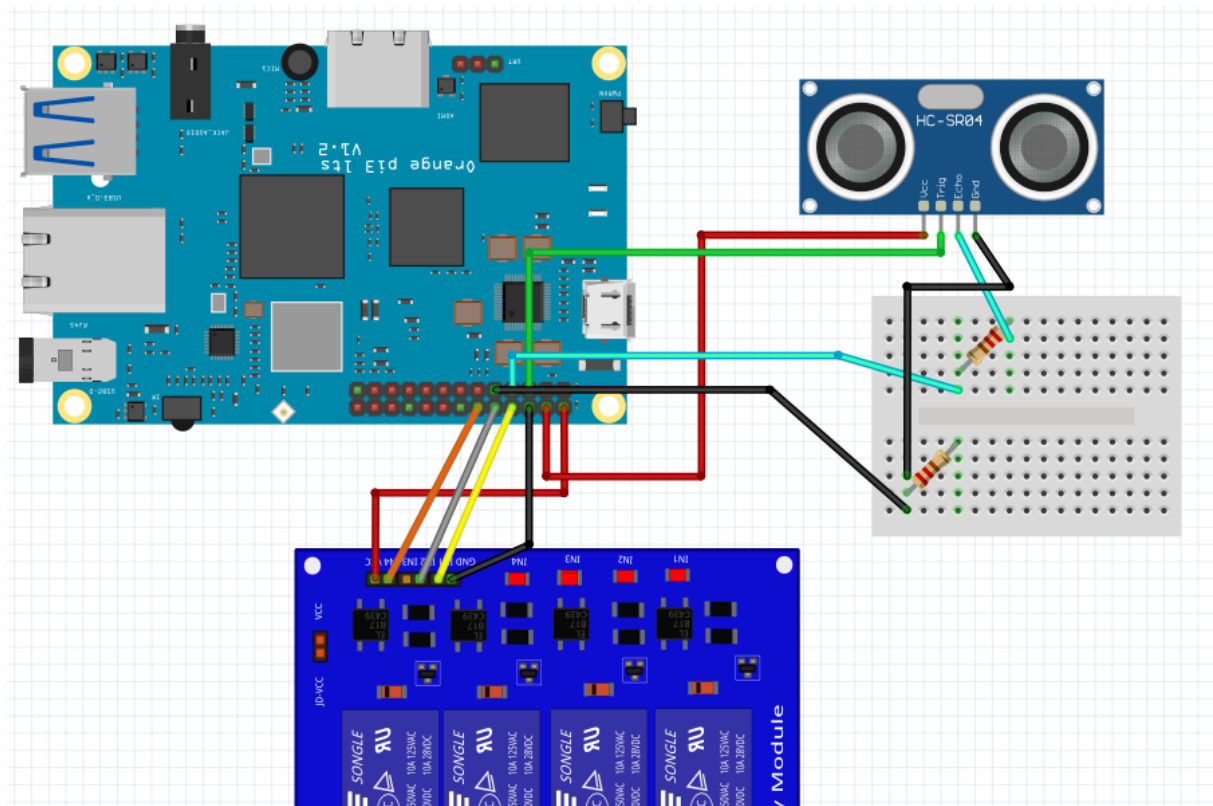


Source	IOT-ESSENTIALS-Kit
Functionalities	- Measure distance between gun and target

The OrangePI constantly keeps track of the distance between the sensor (mounted on top of the gun) and the target. If the target gets too close it sends a signal to the relay, which turns on the smoke machine.

#### 4.2.4.8. Connections (pin-based)

N°	Color schematic	Usage	Source	Destination
1	Red	Vcc	[OPI3] 5V out	[HC] Vcc
2	Lime	Trig	[OPI3] GPIO-1	[HC] Trigger
3	Aqua	Echo	[OPI3] GPIO-2	[HC] Echo
4	Black	GND	[OPI3] GND	[HC] GND

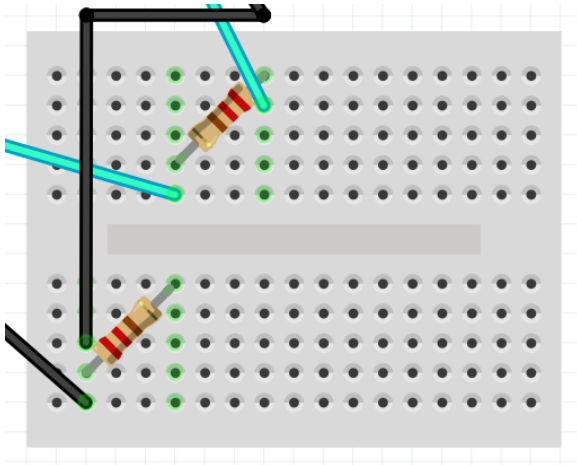


#### 4.2.5. (2x) 220 ohm resistor

##### 4.2.5.1. The component



#### 4.2.5.2. Project integration



They are mounted on a small breadboard where they are used to lower a 5V output coming from the sensor

#### 4.2.6. (1x) LCD 5110

#### 4.2.6.1. The Device



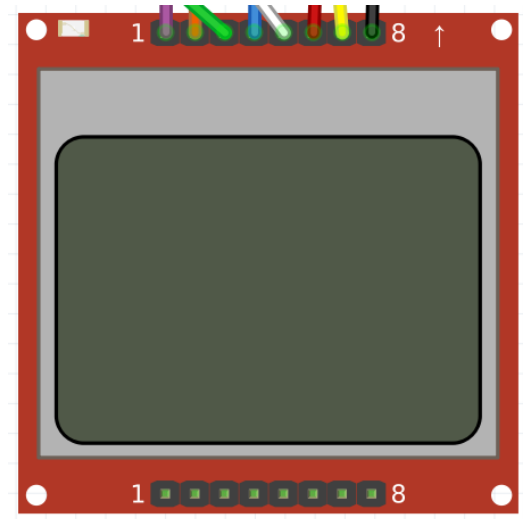
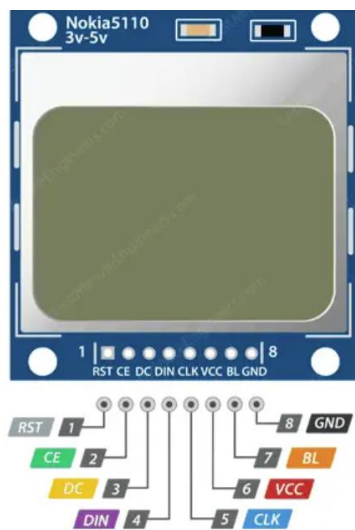
<b>Component</b>	Nokia 5110 LCD Display Module
<b>Manufacturer</b>	Nokia (Controller from Philips)

	Datasheet Controller	<a href="https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf">https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf</a>
	Module Schematic	<a href="https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/Nokia-5110-Module-Schematic.pdf">https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/Nokia-5110-Module-Schematic.pdf</a>

#### 4.2.6.2. Module specifications

<b>Display Technology</b>	Dot Matrix LCD
<b>MCU Interface</b>	SPI
<b>Screen Size</b>	1.5 Inch Across
<b>Resolution</b>	84×48 pixels
<b>Operating Voltage</b>	2.7V – 3.3V
<b>Operating Current</b>	50mA max
<b>Viewing Angle</b>	180°

#### 4.2.6.3. Project integration



<b>Source</b>	IOT-ESSENTIALS-Kit
<b>Functionalities</b>	- Displays interesting Data

*This module is mounted on the side of our device and displays where the AI considers the target is*

#### 4.2.6.4. Connections (pin-based)

N°	Color schematic	Usage	Source	Destination
1	Black	GND	[PICO] GND	[LCD] GND
2	Yellow	GP	[PICO] SPI1 CSn	[LCD] BL
3	Red	3V3	[PICO] 3.3V out	[LCD] VCC
4	White	GP	[PICO] SPI0 SCK	[LCD] CLK
5	Blue	GP	[PICO] SPI0 TX	[LCD] DIN

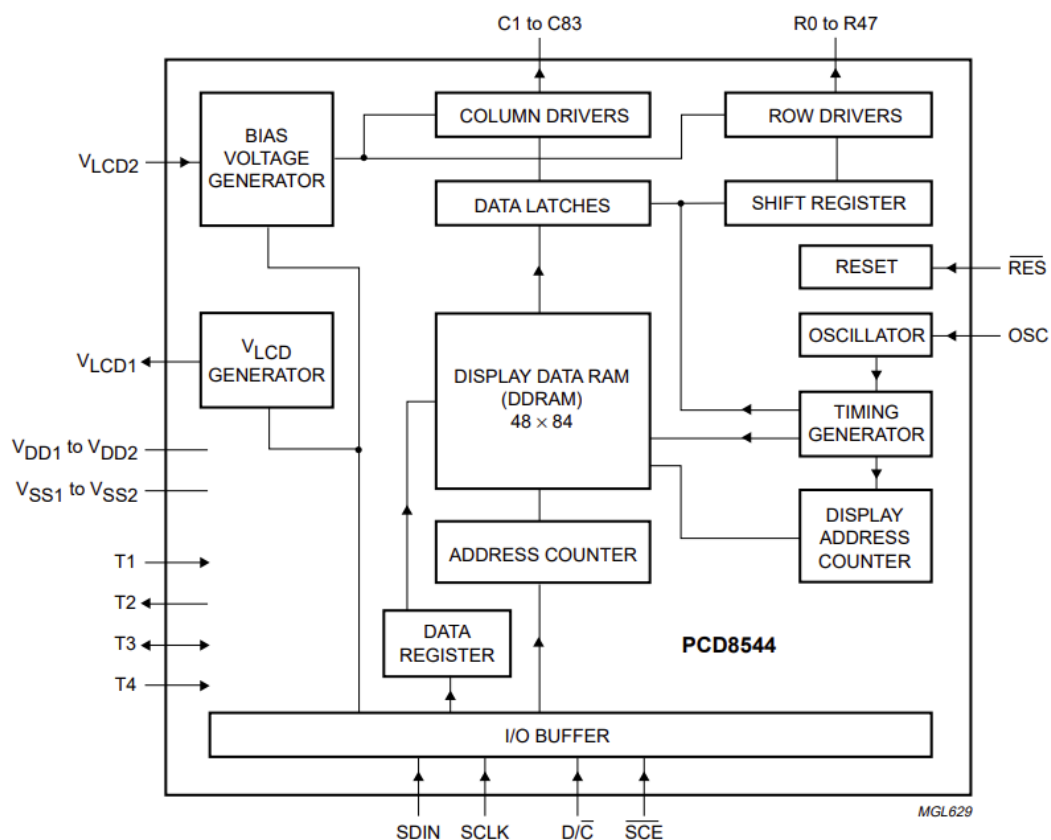
6	Lime	SPIO	[PICO] SPI0 RX	[LCD] DC
7	Orange	SPIO	[PICO] SPI0 CSn	[LCD] CD
8	Purple	GP	[PICO] SPI1 RX	[LCD] RST

- Power supply: 3
- Ground: 1

#### 4.2.6.5. The controller

- PCD8544 (Philips)
- 48x48 pixels matrix LCD controller/driver

#### BLOCK DIAGRAM



#### COMPONENT EXPLANATION

(Source: Datasheet)

#### OSCILLATOR

- Provides clock signal for display
- No need for external clock
  - o Want one still? => OSC connected to Vdd input

### ADDRESS COUNTER

- Assigns addresses to display data RAM for writing
- The X-address X6 to X0
- The Y-address Y2 to Y0 are set separately.
- After a write operation, the address counter is automatically incremented by 1, according to the V flag.

### DISPLAY DATA RAM (DDRAM)

- 48x48 bit
- Static
- 6 banks of 84 bytes (6 x 8 x 48 bits)
- Input: serial interface

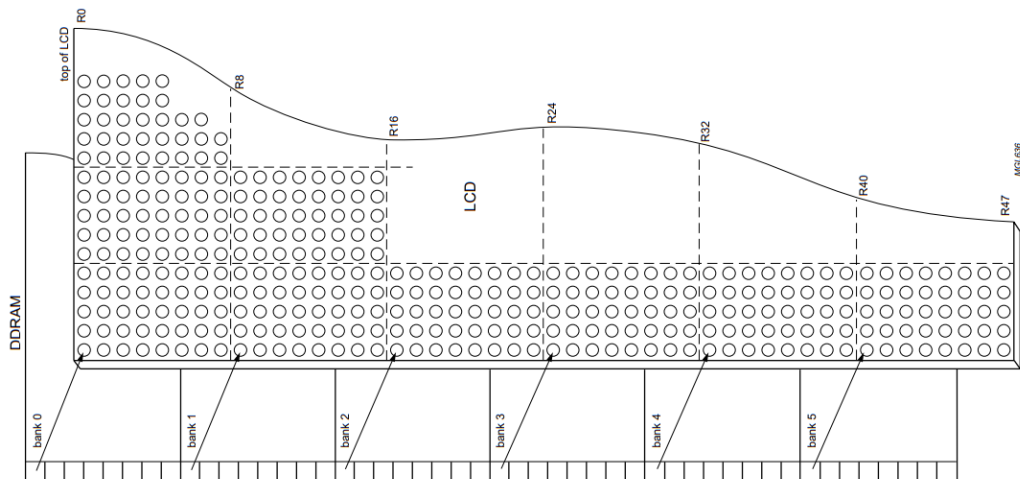
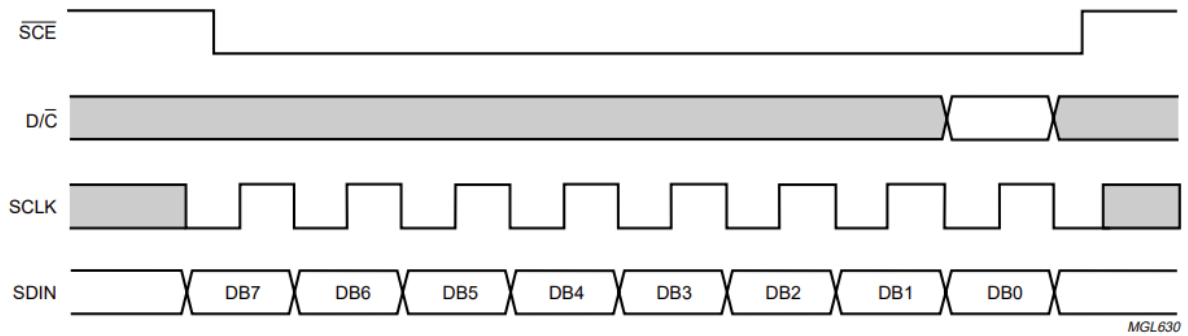


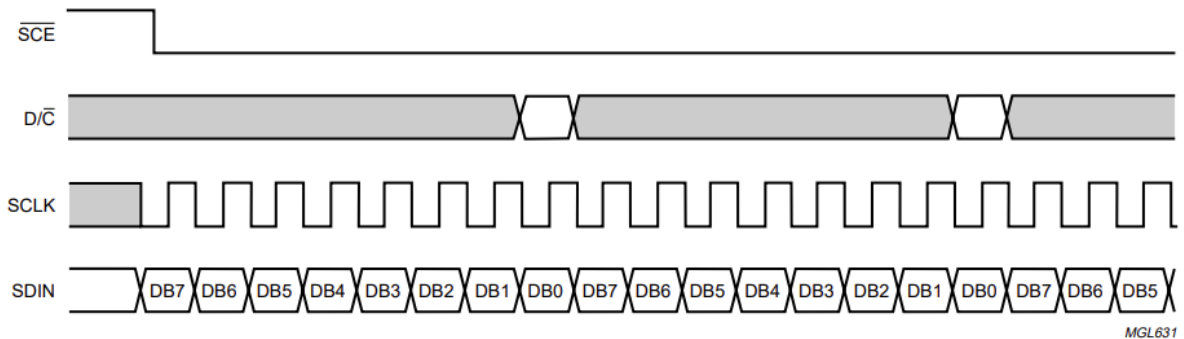
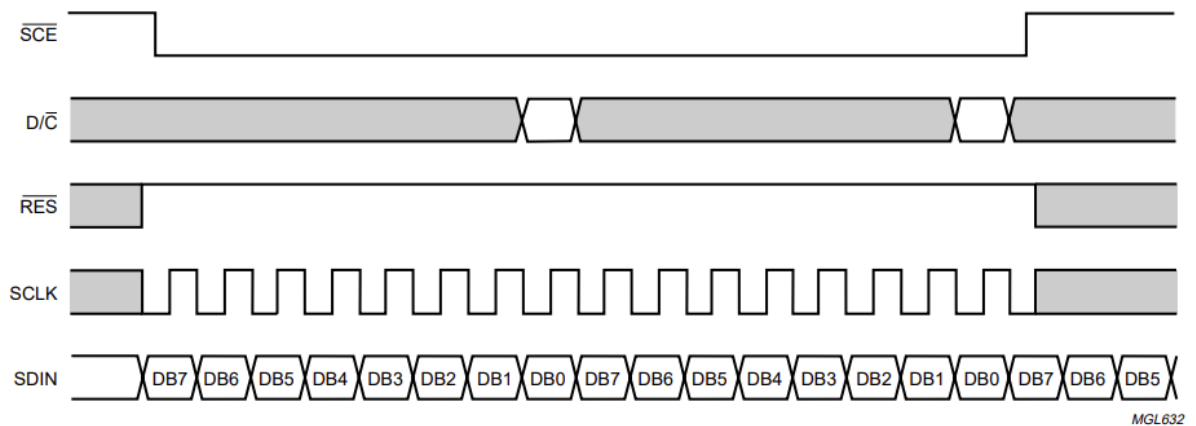
Fig.3 DDRAM to display mapping.

### DISPLAY ADDRESS COUNTER

- Display generated by shifting rows of RAM data to dot matrix LCD

**DATA TRANSMISSION****TRANSMISSION OF ONE BYTE**

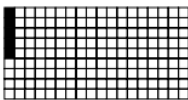
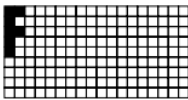
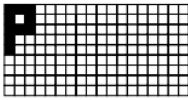
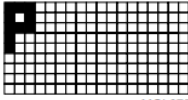

- During transmission: SCE (NOT) is pulled down
- D/C Display control (seen in examples)
- SCLK: clock
- SDIN: data input

**TRANSMISSION OF MULTIPLE BYTES****SERIAL BUS RESET FUNCTIONS**

- Addition of RES line => overwrite active bits

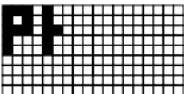
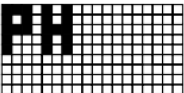

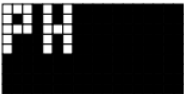



## TRANSMISSION EXAMPLE

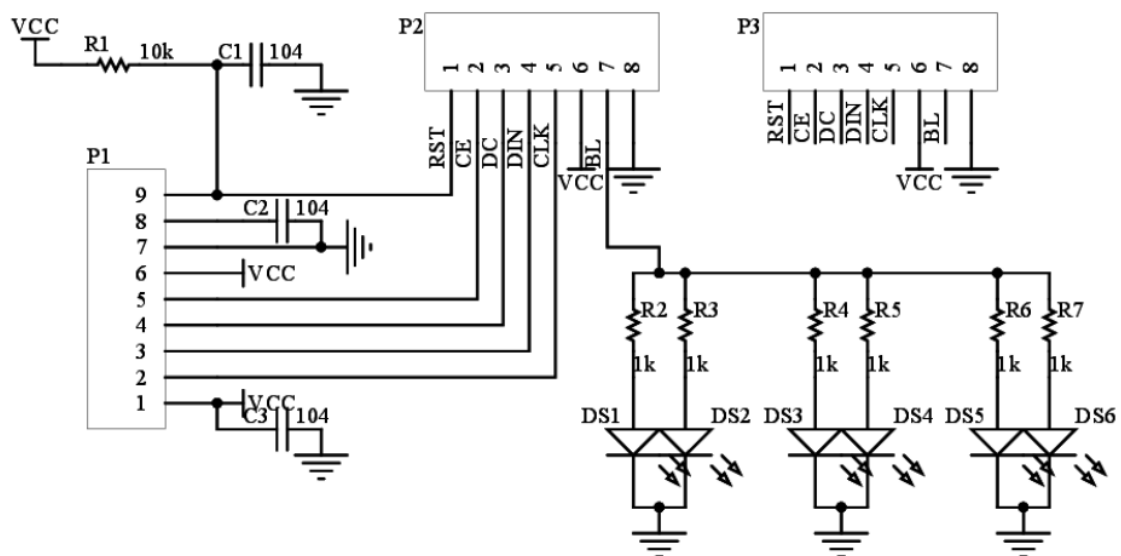
STEP	SERIAL BUS BYTE									DISPLAY	OPERATION
	D/C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
1	start										$\overline{\text{SCE}}$ is going LOW
2	0	0	0	1	0	0	0	0	1		function set PD = 0 and V = 0, select extended instruction set (H = 1 mode)
3	0	1	0	0	1	0	0	0	0		set $V_{OP}$ ; $V_{OP}$ is set to a $+16 \times b [V]$
4	0	0	0	1	0	0	0	0	0		function set PD = 0 and V = 0, select normal instruction set (H = 0 mode)
5	0	0	0	0	0	1	1	0	0		display control set normal mode (D = 1 and E = 0)
6	1	0	0	0	1	1	1	1	1	 MGL673	data write Y and X are initialized to 0 by default, so they are not set here
7	1	0	0	0	0	0	1	0	1	 MGL674	data write
8	1	0	0	0	0	0	1	1	1	 MGL675	data write
9	1	0	0	0	0	0	0	0	0	 MGL675	data write
10	1	0	0	0	1	1	1	1	1	 MGL676	data write

- Mark: empty line between P and I

## DISPLAY CONTROL EXAMPLE

STEP	SERIAL BUS BYTE									DISPLAY	OPERATION
	D/C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
11	1	0	0	0	0	0	1	0	0	 MGL677	data write
12	1	0	0	0	1	1	1	1	1	 MGL678	data write
13	0	0	0	0	0	1	1	0	1	 MGL679	display control; set inverse video mode (D = 1 and E = 1)
14	0	1	0	0	0	0	0	0	0	 MGL679	set X address of RAM; set address to '0000000'
15	1	0	0	0	0	0	0	0	0	 MGL680	data write

#### 4.2.6.6. Component schematic



## 4.2.7. (1x) Watergun

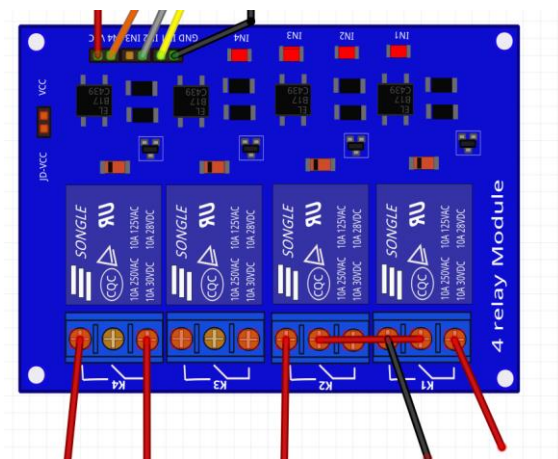
### 4.2.7.1. The object



### 4.2.7.2. Project integration

How do we trigger the gun with a relay?

- Trigger closed permanently
- Relay put between battery and gun
  - o When turned on => gun activated



R2 is left open to activate the remote power

R1 is used to switch between button A and B

### 4.2.8. (5x) L298N Motor Driver

#### 4.2.8.1. The object



1.1.1. (5x) L298N Motor Driver



L298

<b>Component</b>	L298N Dual H-Bridge Motor Driver
<b>Manufacturer</b>	Handson Technology

	L298N Controller	<a href="https://www.mouser.be/datasheet/2/389/l298-1849437.pdf">https://www.mouser.be/datasheet/2/389/l298-1849437.pdf</a>
	Motor Driver	<a href="https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf">https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf</a>

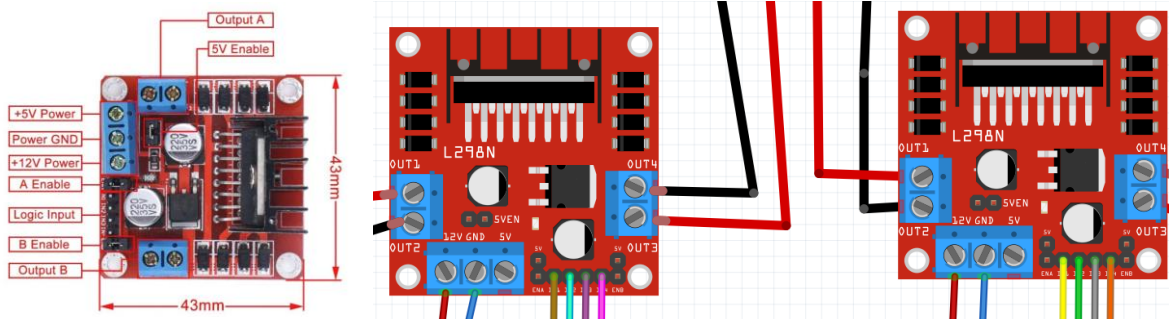
#### 4.2.8.2. Module specifications

Parameter	Value
Input Voltage	3.2V ~ 40Vdc
Driver	L298N Dual H Bridge DC Motor Driver
Power Supply	DC 5 V - 35 V
Peak Current	2 Amp
Operating Current Range	0 ~ 36mA
<b>Control Signal Input Voltage Range</b>	
Low	$-0.3V \leq V_{in} \leq 1.5V$
High	$2.3V \leq V_{in} \leq V_{ss}$
<b>Enable Signal Input Voltage Range</b>	
Low	$-0.3 \leq V_{in} \leq 1.5V$ (control signal is invalid)
High	$2.3V \leq V_{in} \leq V_{ss}$ (control signal active)
Maximum Power Consumption	20W (when the temperature $T = 75^{\circ}C$ )
Storage Temperature	$-25^{\circ}C \sim +130^{\circ}C$
On-board +5V Regulated Output Supply	Yes (supply to controller board e.g., Arduino)
Size	3.4cm x 4.3cm x 2.7cm

#### 4.2.8.3. Project Integration

Source	Ordered online
Functionalities	<ul style="list-style-type: none"> <li>- Operate stepper motors</li> <li>- Give them required voltage</li> </ul>

*This driver is used to operate the two stepper motors that move the gun and the entire device on both the X-axis and the Y-axis. The driver is operated by the Raspberry PI Pico.*

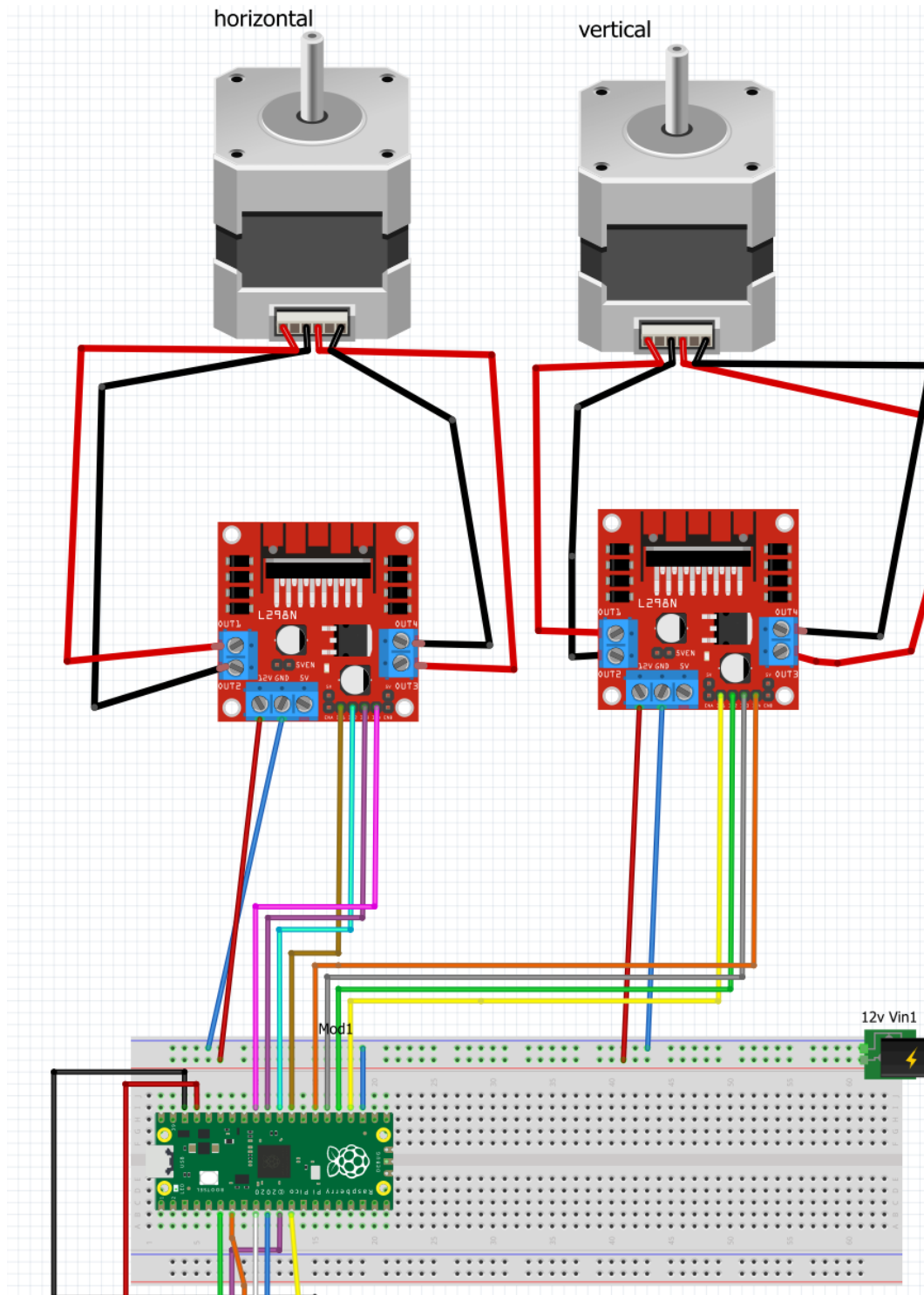


*The left one is for horizontal movement, the right on is for vertical movement.*

#### 4.2.8.4. Connections (pin-based)

N°	Color schematic	Usage	Source	Destination
1	Brown [H]	GPIO	[PICO] GP-22	[L298N] (H) IN 1
2	Aqua [H]	GPIO	[PICO] RUN	[L298N] (H) IN 2
3	Purple [H]	GPIO	[PICO] GP-26	[L298N] (H) IN 3
4	Pink [H]	GPIO	[PICO] GP-27	[L298N] (H) IN 4
5	Yellow [V]	GPIO	[PICO] GP-18	[L298N] (V) IN 1
6	Lime [V]	GPIO	[PICO] GP-19	[L298N] (V) IN 2
7	Gray [V]	GPIO	[PICO] GP-20	[L298N] (V) IN 3
8	Orange [V]	GPIO	[PICO] GP-21	[L298N] (V) IN 4
9	Red [H]	12V input	[BAT] 12V input	[L298N] (H) 12V
10	Blue [H]	GND	[PICO] GND	[L298N] (H) GND
11	Red [V]	12V input	[BAT] 12V input	[L298N] (V) 12V
12	Blue [V]	GND	[PICO] GND	[L298N] (V) GND

- Horizontal
  - o Connections to control steps: 1, 2, 3, 4
  - o Power supply: 9
  - o Ground: 10
- Vertical
  - o Connections to control steps: 5, 6, 7, 8
  - o Power supply: 11
  - o Ground: 12



#### 4.2.8.5. L298 Controller

<a href="#">L298N Controller</a>	<a href="https://www.mouser.be/datasheet/2/389/l298-1849437.pdf">https://www.mouser.be/datasheet/2/389/l298-1849437.pdf</a>
----------------------------------	---

#### WHY DO WE NEED THESE?

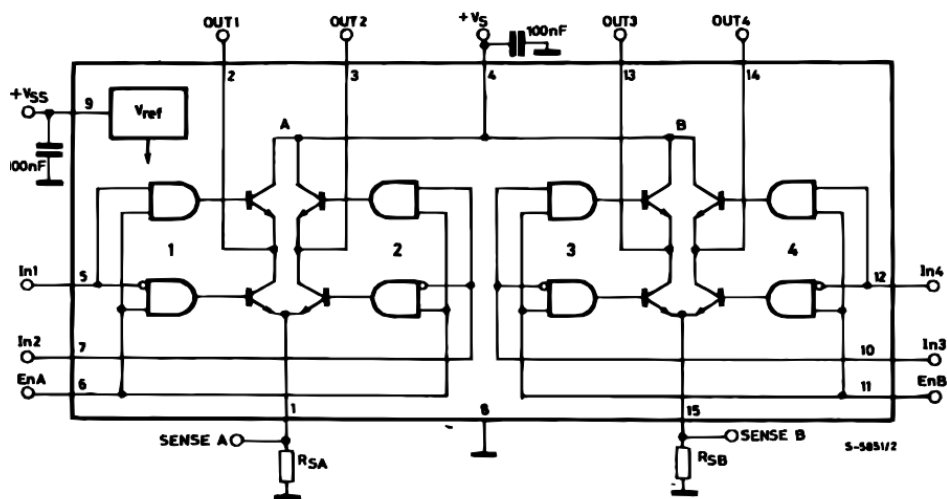
- Motors require higher voltage rating then our boards are compatible with (12V for each)
- Current rating to high

#### SPECIFICATIONS OF BOARD

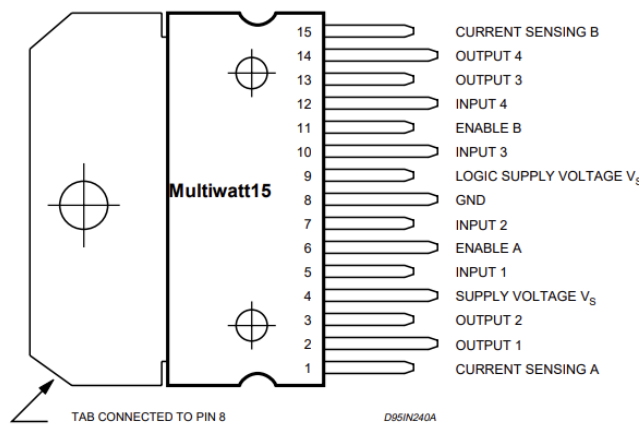
Spec	Value
Max U (V)	46V
Max I (A)	4A
Extra	<ul style="list-style-type: none"> <li>- Overtemperature protection</li> <li>- High noise immunity</li> </ul>

*"The L298 is an integrated monolithic circuit in a 15- lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors"*  
(Definition as given in the datasheet)

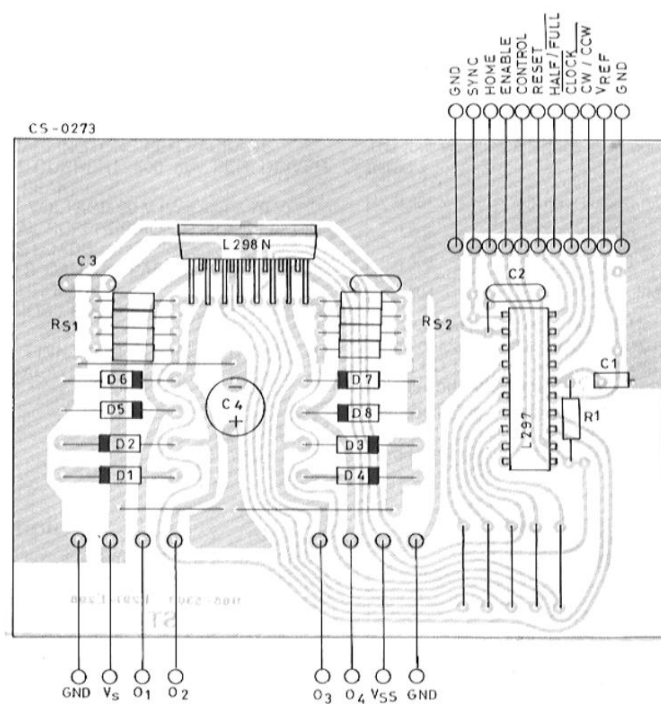
#### BLOCK DIAGRAM



## PIN CONNECTIONS



## CIRCUIT BOARD



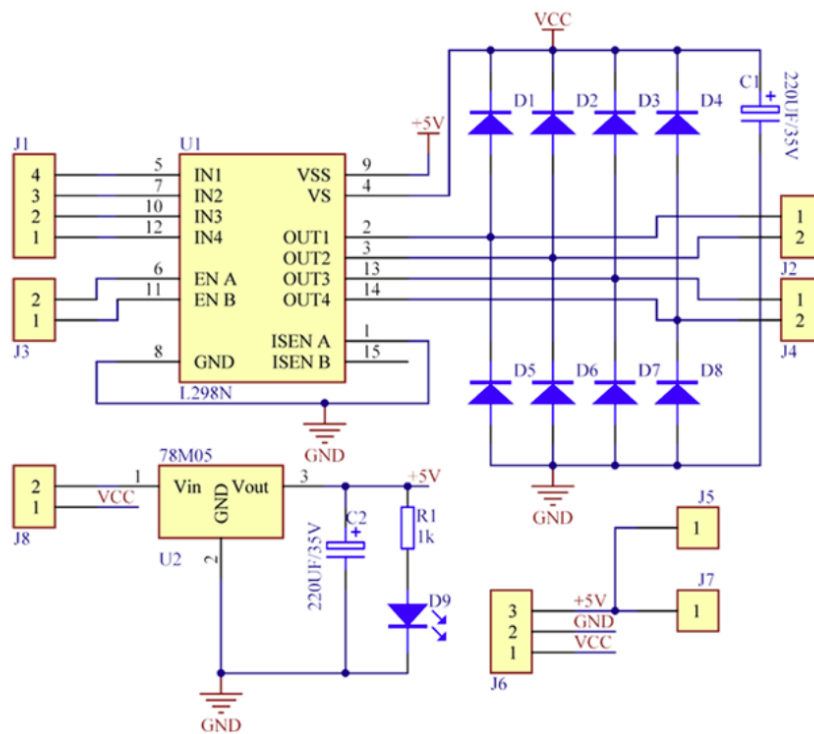
The datasheet does not provide an explanation on how this works exactly.

*Phases will be mentioned in the Nema 17 section*

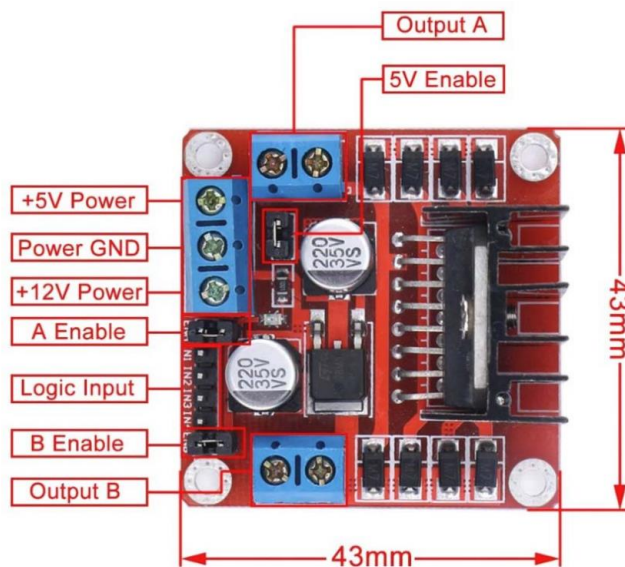


#### 4.2.8.6. L298N Dual H-Bridge Motor Driver

##### SCHEMATIC

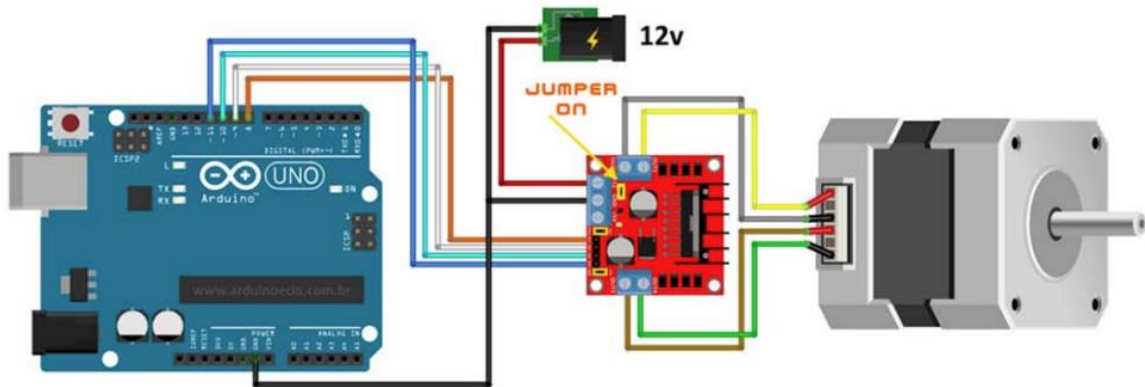


##### PIN FUNCTIONS



## HOW DOES IT WORK?

### EXAMPLE WITH ONE MOTOR



The OrangePI (Arduino Uno in this example) Uses a program that provides the right sequence of steps through the four cables (blue, aqua, white, orange). The 12V power supply is hooked up to the motors. And the motor is connected to the two sides of the driver. Note that the driver can also be used to drive two DC motors, which is why the connections are on either side of the driver.

*The program part has (hopefully) been explained in the code section*

## 4.2.9. (1x) Smoke machine

### 4.2.9.1. The Product



### 4.2.9.2. Project Integration

When a person comes to close, the distance sensor will detect that and send a signal to the OrangePI, which will turn on the relay and cover the enemy with smoke, similar to how modern security protects stores from burglars.

🔗	Example: NUBI 4.0	<a href="https://www.smarteksrl.it/en/home/">https://www.smarteksrl.it/en/home/</a>
---	-------------------	---



#### 4.2.9.3. How do we trigger the smoke machine?

- Hotwire the remote for the machine
  - o One switch for turning on remote power
  - o One switch for deciding whether to start or stop the flow of smoke

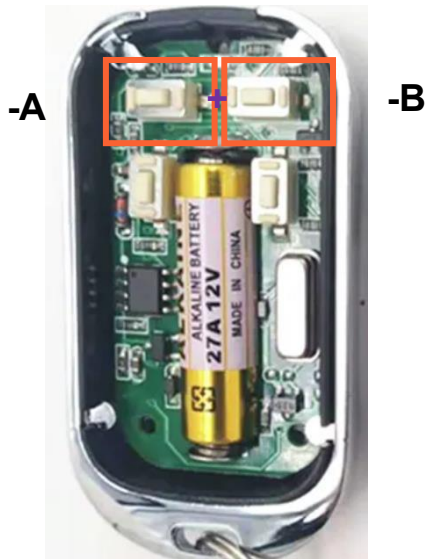
#### THE REMOTE



## THE BUTTONS

- Letting electricity flow through the top left (A) button turns on the smoke
- Letting electricity flow through the top right (B) button turns off the smoke

## HOW DO WE HOTWIRE THIS?



We attach a cable to the positive terminal at the center and two wires to both the negative terminal on the left and the negative terminal on the right.

- Relay 1:

Left	Common	Right
Cable -A	Cable C	Cable -B

- Relay 2:

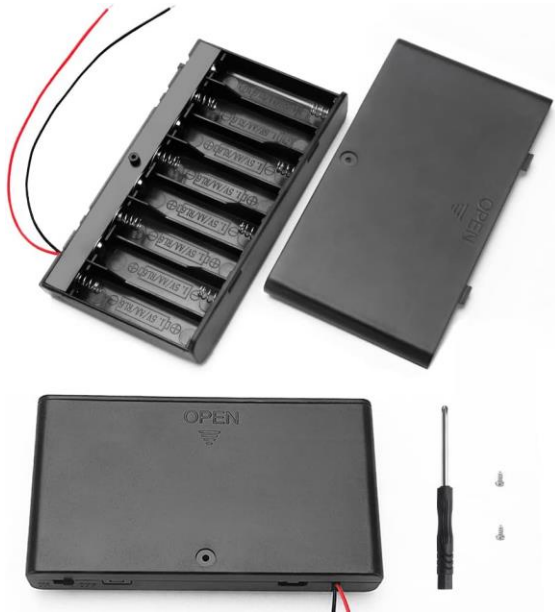
Left	Common	Right
Power supply	Cable C	Empty

- The first relay will decide the button that is powered on
- The second relay will turn on the power

*The second button setup is added to make sure that the power is not always enabled, to make sure that the battery lasts longer*

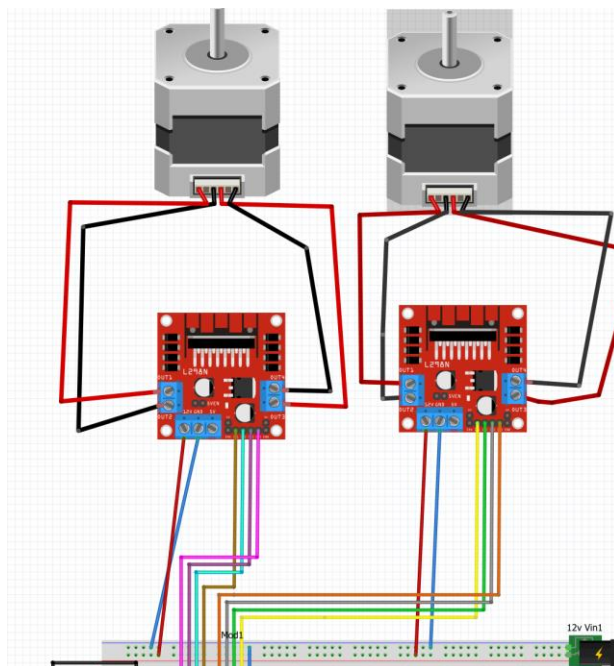
#### 4.2.10. (2x) 12V Battery holders

##### 4.2.10.1. The object



##### 4.2.10.2. Project Integration


- Batteries are used as a 12V power supply for the motors



### 4.2.11. (5x) Nema 17

#### 4.2.11.1. The product

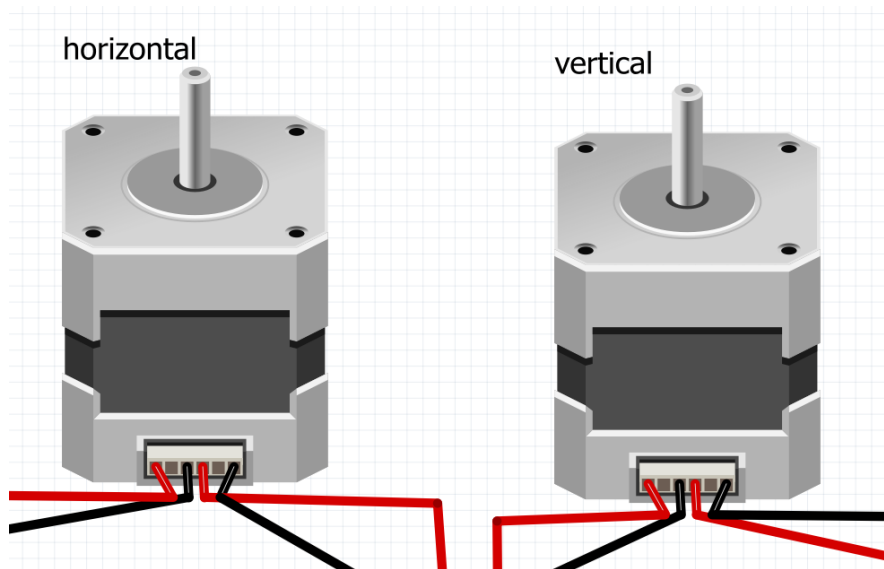


	Datasheet	<a href="https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf">https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf</a>
---	-----------	---

#### 4.2.11.2. Project Integration

Source	Ordered online
Functionalities	- Turns components in order to aim

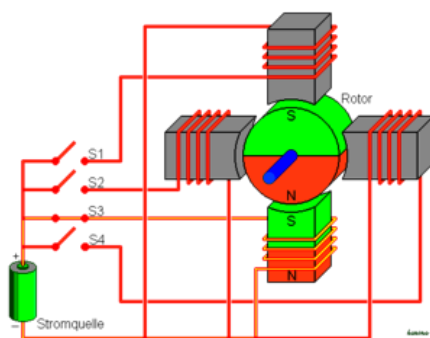
Connected to components using shaft couplers (4.2.12)



#### 4.2.11.3. Motor specifications

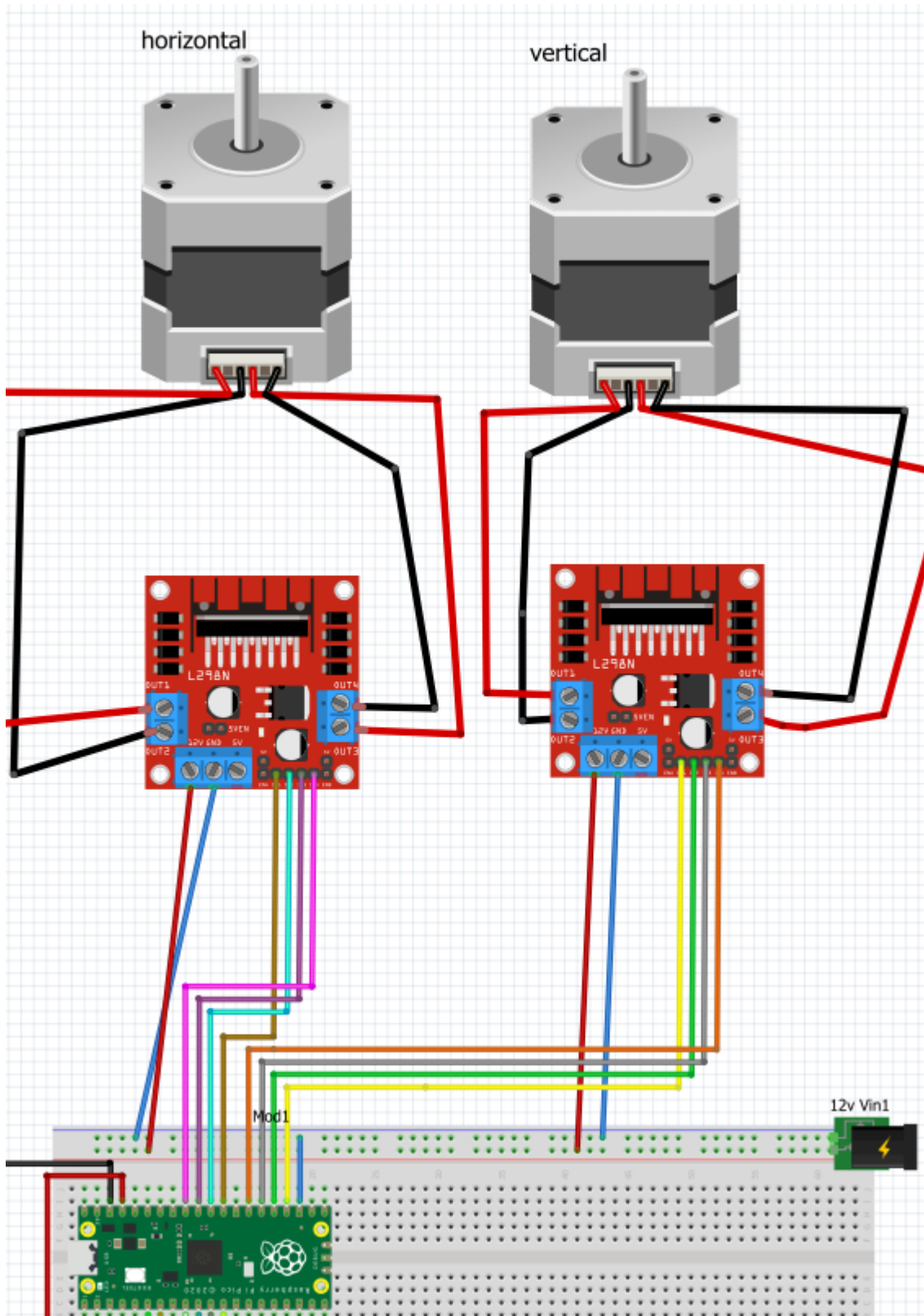
Parameter	Value
Phases	2
Steps/Revolution	200
Step Accuracy	±5%
Shaft Load	20,000 Hours at 1000 RPM
Axial Load	
Push	25 N (5.6 lbs.)
Pull	65 N (15 lbs.)
Radial Load	29 N (6.5 lbs.) At Flat Center
IP Rating	40
Approvals	RoHS
Operating Temperature	-20° C to +40° C
Insulation Class	B, 130° C
Insulation Resistance	100 Mohm

- 2 Phase





#### 4.2.11.4. How do we control them?



Connected to the Pico using the two L298N drivers



## 4.2.12. (4x) Shaft coupler

### 4.2.12.1. The object



### 4.2.12.2. Project Integration

- Used to attach motors to the rotating structural components

*More information in Section 5*

## 5. Physical structure

### 5.1. Software to make parts

#### 5.1.1. FreeCAD

*Freecad is an open source modeling software program. Made with the intent to make 3d modeling for objects of any size or complexity free, easy and more comprehensive than other more known CAD modeling programs out there.*

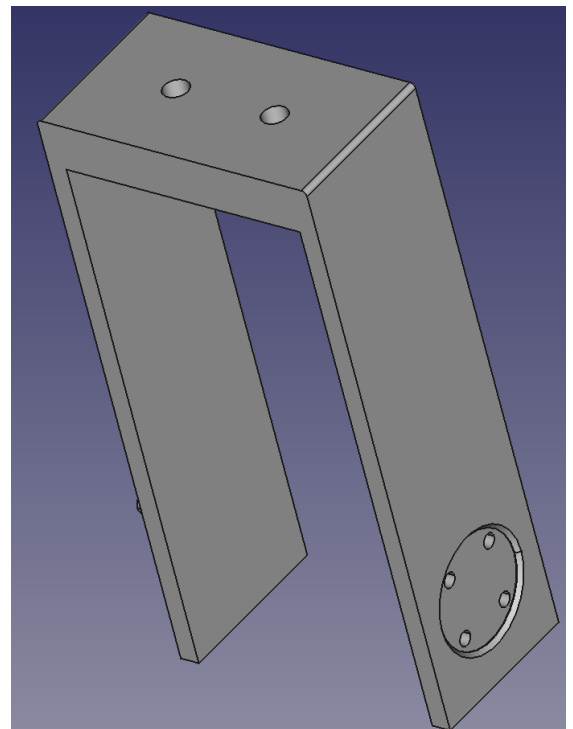


### 5.2. 3D Printed components

#### 5.2.1. Main Gun mount

This mount is attached the 2 sides of the Gun Top using the Motor Stem and a Ball Bearing.

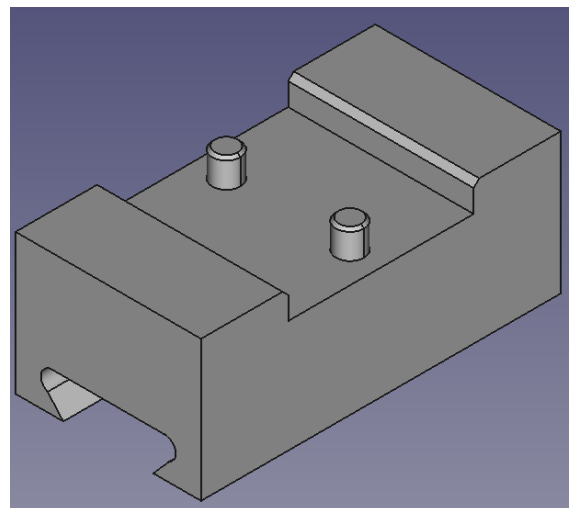
This has been done to make sure that the whole mount can spin freely alongside it when the motor receives a signal



#### 5.2.2. Gun Barrel mount

Originally the gun came with a scope attachment. Which we ended up redesigning and 3d printing it.

After doing this we are able to use this to attach the Main Gun Mount to the barrel and in turn make the gun rotate on its Y-axis without having to dismantle the gun in any way.

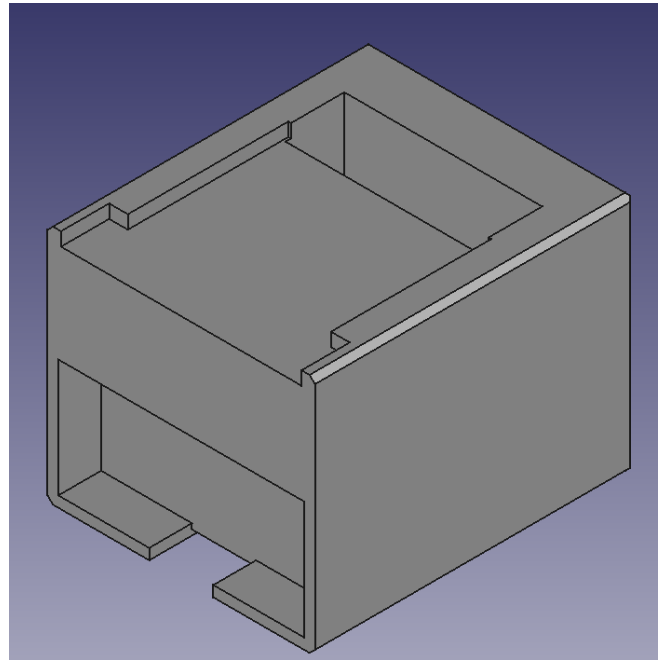


### 5.2.3. Camera and distance sensor mount

Like the gun barrel mount this component is also attached to the Main Gun Mount. This part is designed to hold the camera, which can therefore be held into place easily.

The front of the mount also contains a spot for the distance sensor used for the smoke machine.

Ideally in this way the camera and sensor would be able to see whatever is in front and how far from the gun to aim accordingly.



### 5.2.4. Gun Top

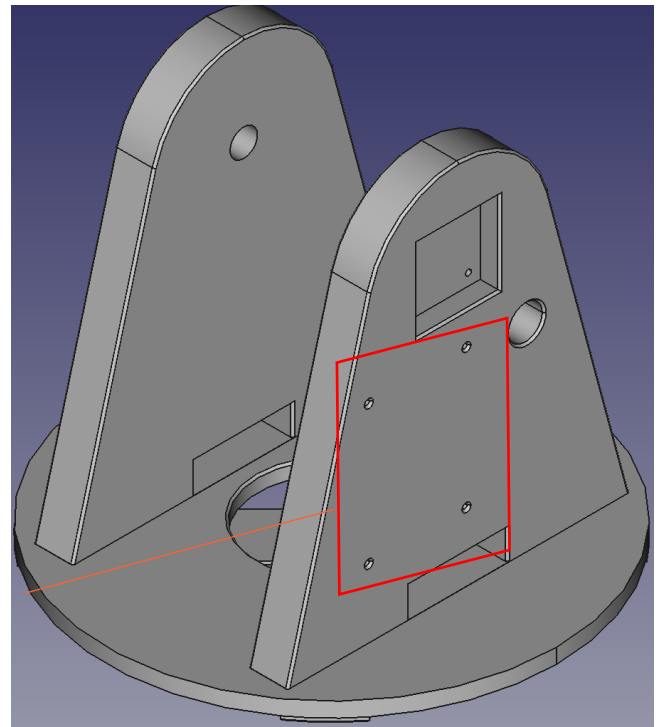
This mount is attached to the bottom motor which in turn spins the Mount and gun on their X-axis.

The bottom of the mount made in an X shape on order to ideally better redistribute the spinning force from the small attachment to the motor stem to the large circular mount.

It also contains multiple holes used for cable management.

These hole would be located at the middle, one at the bottom of each side, and a circular whole at the side of the motor.

At the very top of the arch there is a hole for the ball bearing and for screwing in the motor and on the side you can also find the mount for the relay



### 5.2.5. Gun bottom

The bottom is made into a circular shape with one 90 degree edge. Which is used for the battery pack and breadboard placement.

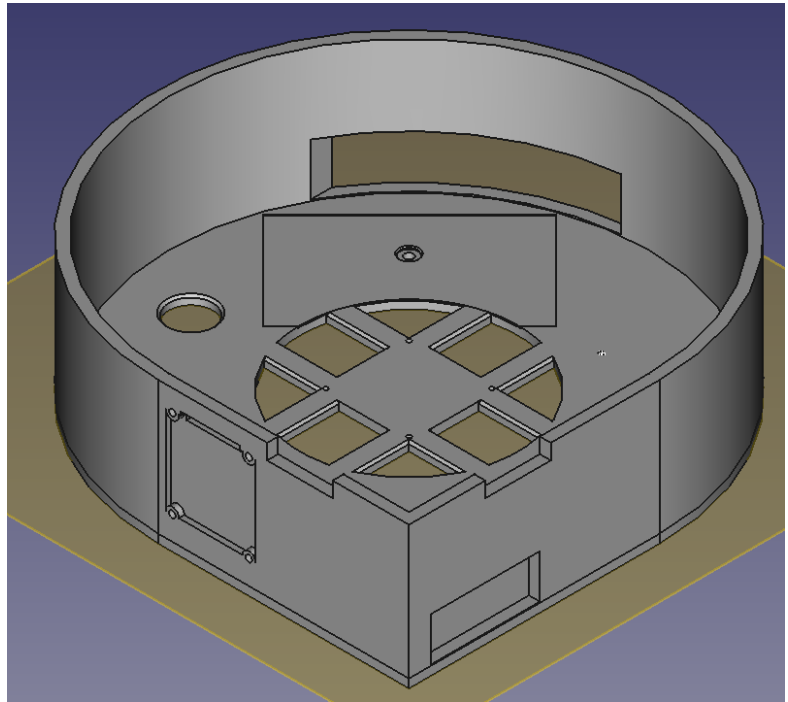
All along the side and bottom there are cutouts for multiple reasons.

Which would be screw holes, a ventilation hole in the middle, cutouts for cables and a cutout for the on and off switch.

In the very middle the bottom used to spin the top is placed. To the side of it there are both the motor controllers.

At the side near the power switch there also is a mount for the screen placements.

At the very bottom in the middle there is a fan we may or may not use. The bottom also has feet which we placed for the sake of vibration reduction and for giving the fan more space to breath.



## 6. Video

<https://www.youtube.com/watch?v=4RwN2gXnb0c>

## 7. Self evaluation

Item	Max	My score	Motivation
Original concept	2	2	Self-defense against other team from class
Interfaces	8	5	
GPIO, Stepper, PWM	(3)	2	For the rotation of the gun, pico, motor, etc.
SPI, Analog in, LCD	(3)	3	For the display of the status of the system and distance.
I2C	(2)	0	
MQTT via Pico	4	4	For the communication between Pi-Pico
Dashboard	2	0	
Extra: AI, ...	2	2	Recognize the position/presence of intruder
Nice demo / video / report	2	2	Good and detailed documentation and demonstration
<b>Total</b>	<b>20</b>	<b>17</b>	

## 8. References

### 8.1. DataSheets / User manuals

Component	Link
Raspberry Pi Pico	<a href="https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf">https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf</a>
OrangePI 3 LTS	<a href="https://uelectronics.com/wp-content/uploads/2022/04/OrangePi-3-LTS-H6-User-manual-v1.0.pdf">https://uelectronics.com/wp-content/uploads/2022/04/OrangePi-3-LTS-H6-User-manual-v1.0.pdf</a>
HCSR04	<a href="https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf">https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf</a>
LCD 5110 Controller	<a href="https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf">https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf</a>
L298N Dual H-Bridge Motor Driver	<a href="https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf">https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf</a>
L298	<a href="https://cdn.bodanious.com/media/1/9d61596_img.pdf">https://cdn.bodanious.com/media/1/9d61596_img.pdf</a>
Nema 17	<a href="https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf">https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf</a>

### 8.2. Schematics

Component	Link
LCD 5110	<a href="https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/Nokia-5110-Module-Schematic.pdf">https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/Nokia-5110-Module-Schematic.pdf</a>

### 8.3. Other

- Original HCSR04 module: <https://github.com/rsc1975/micropython-hcsr04/tree/master>
- FreeCAD Tutorial: <https://www.youtube.com/watch?v=rglvJH9z5ng>



## CONTACT

Naam | Functie  
xxx.xxx@thomasmore.be  
Tel. + 32 xx xx xx xx

## VOLG ONS

www.thomasmore.be  
fb.com/ThomasMoreBE  
#WeAreMore

THOMAS  
**MORE**