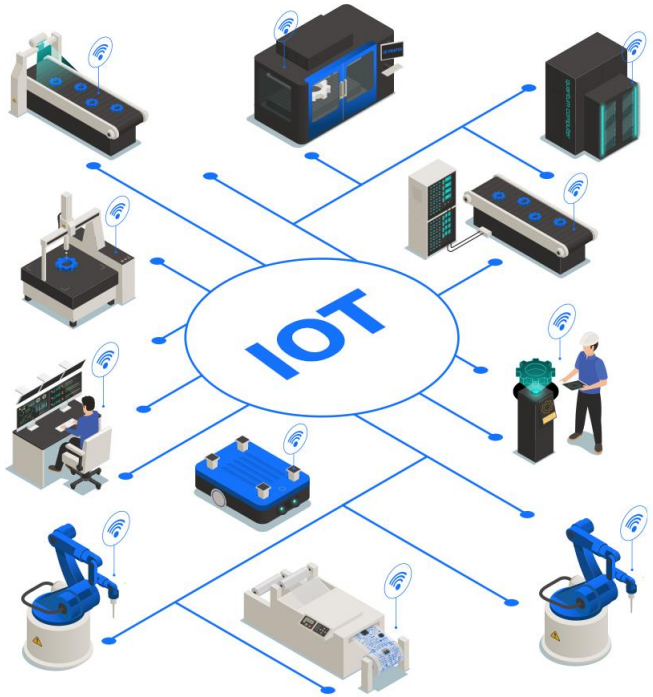


Industrial IoT. Image Source: IoT Innovations | Top 33 Industrial IoT Device Examples (bytebeam.io).

/ SS2024_AES_Lab_TeamC

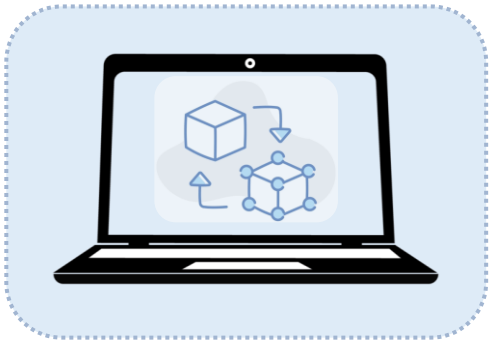
Teammates (alphab.):
- Akram Md
- Richard Jimenez

Module: Advanced Embedded Systems – SuSe24
Examiners: Prof. Dr. Hayek
ELE – Hochschule Hamm-Lippstadt
July 2nd 2024

IoT Graphical Block Diagram

IIOT APPLICATION

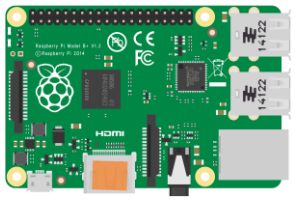
Source: Own Creation.



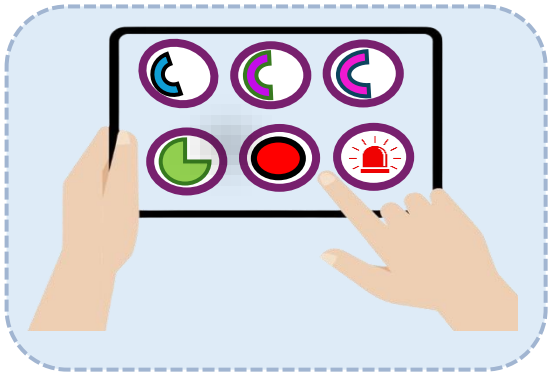
MQTT Client

Subscribe

Subscribe



MQTT Broker



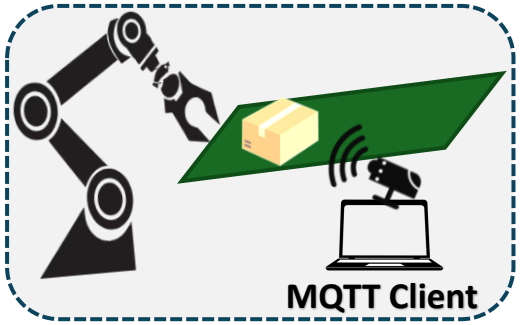
MQTT Client

Publish

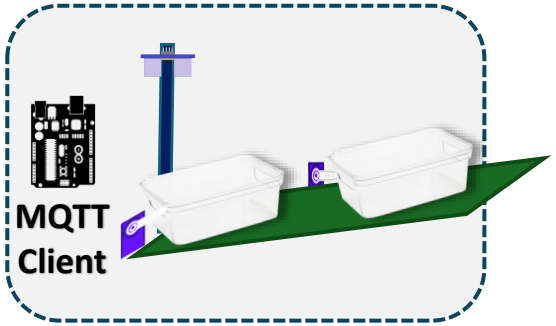
Publish

Publish

Publish

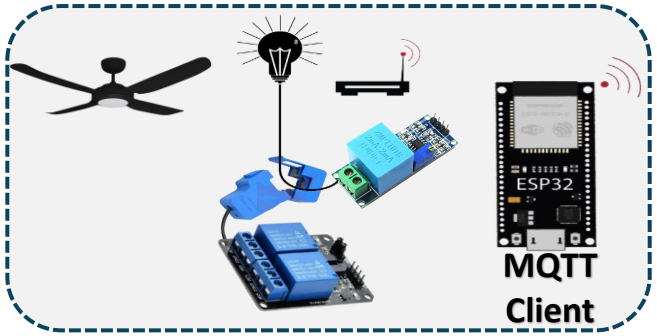


Quality Control



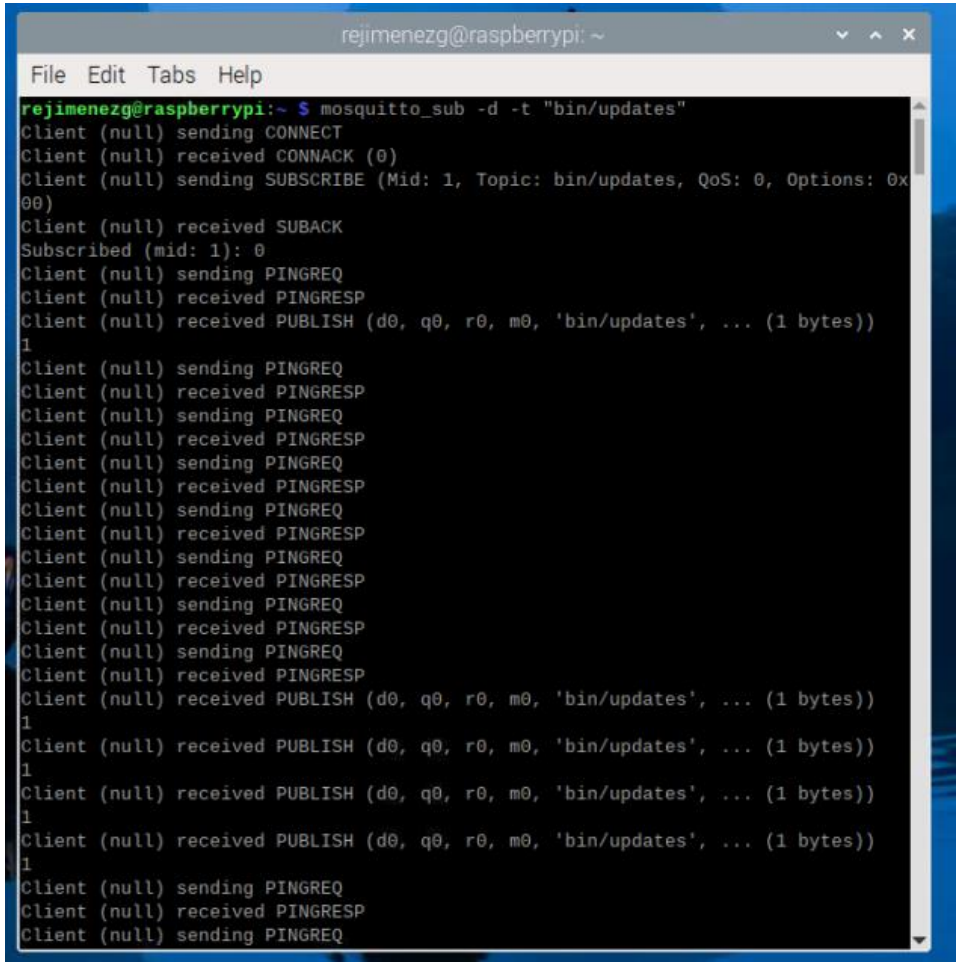
Stock Management

Safety and Security system



Power Monitoring and controlling system

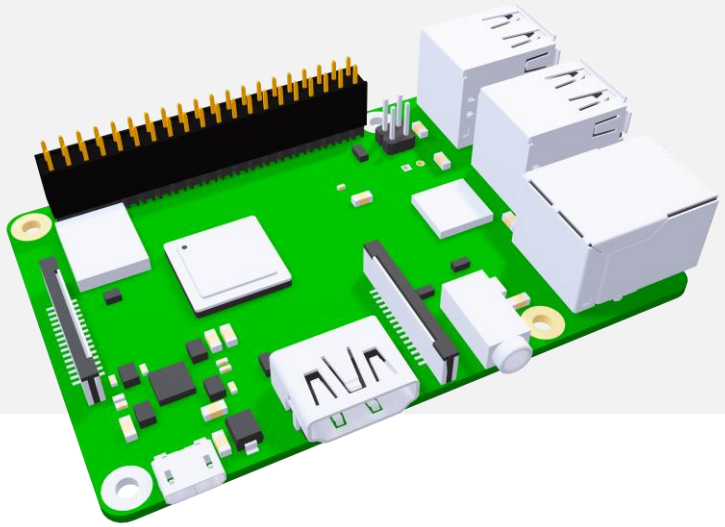
IIoT – MQTT Broker



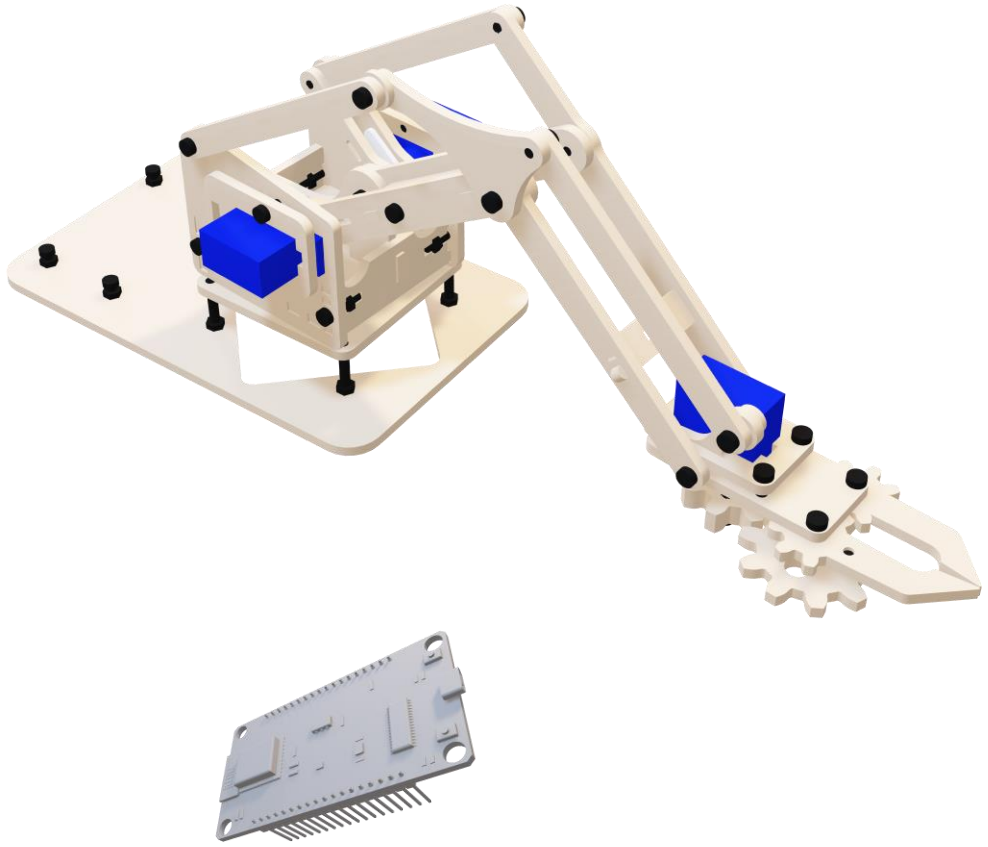
Mosquitto MQTT Broker:

- Raspberry Pi 3

Tools and SW: Mosquitto MQTT, VNCC, PuTTY

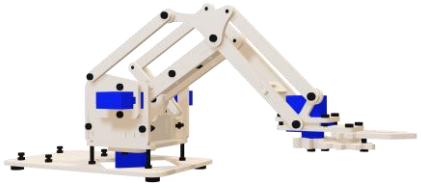


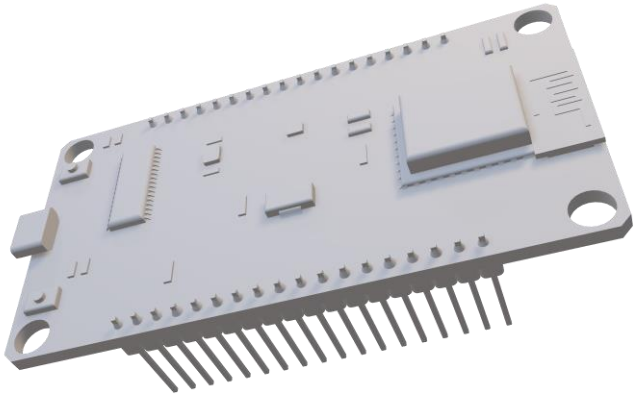
IIoT – Quality Control



- Embedded control system with IIoT integration: Real-time servo control, data analytics, and remote operation capabilities

IIoT – Quality Control

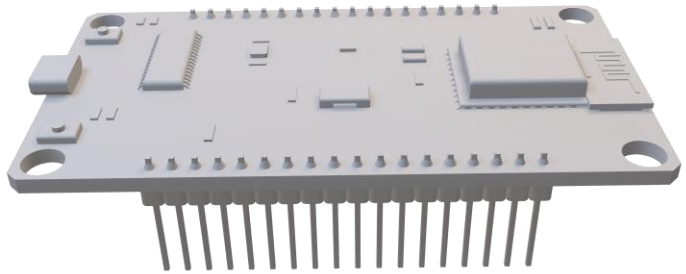


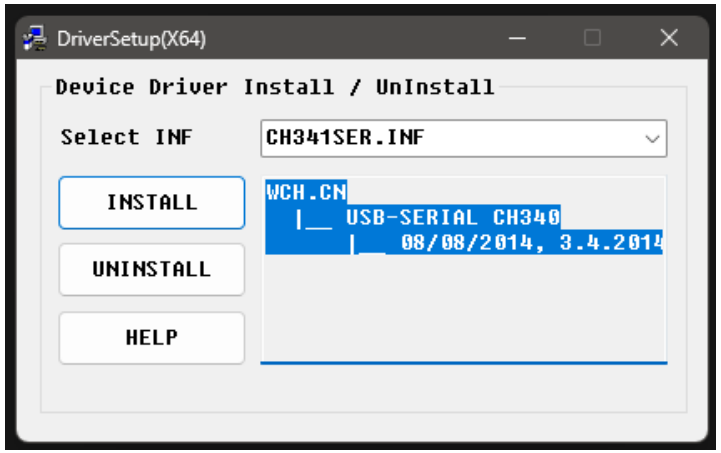


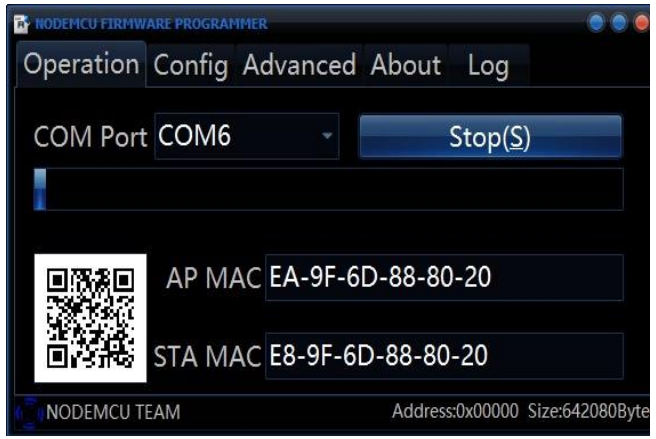
- Embedded control system with IIoT integration: Real-time servo control, data analytics, and remote operation capabilities

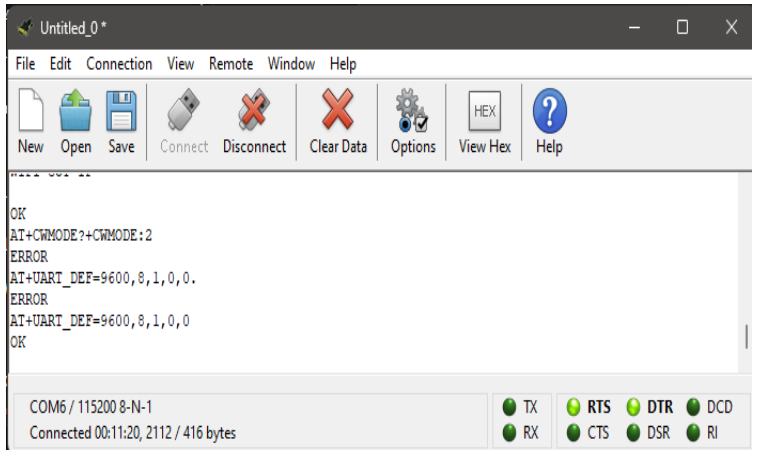
- ESP8266 microcontroller with MicroPython: Wi-Fi-enabled, low-cost solution for IoT connectivity and rapid prototyping

ESP8266 Extra Setup



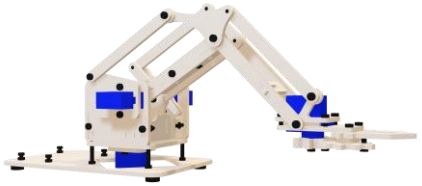


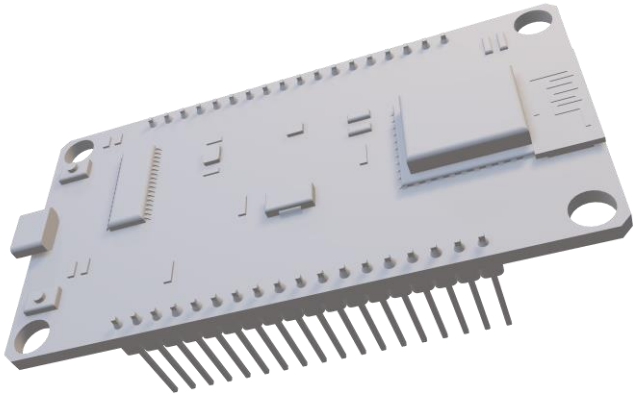


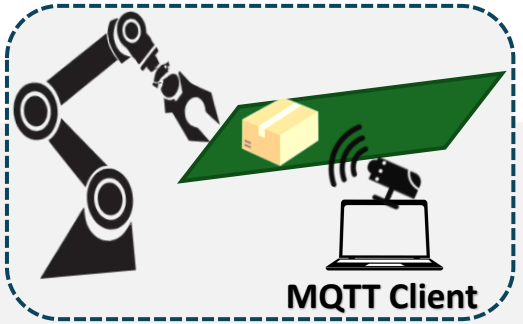


Source: Own Creation.

IIoT – Quality Control





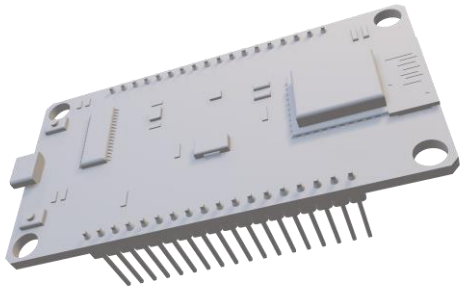


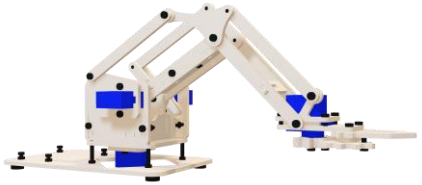
Quality Control

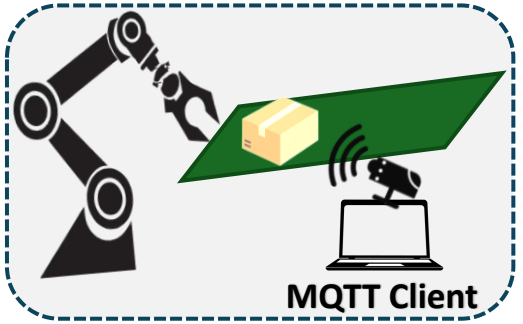
Development Tools for ESP8266 and MicroPython:

- VSCode and Pymakr extension
- PyCharm and MicroPython extension
- uPyCraft IDE

IIoT – Quality Control







Quality Control

Three sources files:

Source: Own Creation.

IIoT – Quality Control Main Sources Code

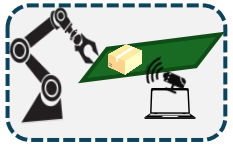
HOCHSCHULE HAMM-LIPPSTADT

boot.py 5, U

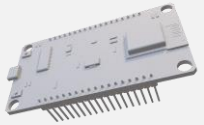
Final_Project_sources > RoboticarmSources > boot.py > ...

```python
import time
from umqttsimple import MQTTClient
import ubinascii
import machine
import micropython
import network
import esp
esp.osdebug(None)
import gc
gc.collect()

ssid = 'aeshotspot' #'UPCBF3B25D'
password = 'aeshotspot' #'Jmhpsapyv23j'
mqtt_server = '192.168.0.59' #
mqtt_user = ''
mqtt_pass = ''

#EXAMPLE IP ADDRESS
#mqtt_server = '192.168.1.144'
client_id = ubinascii.hexlify(machine.unique_id())
topic_sub = b'bin/updates'
topic_pub = b'bin/updates'

last_message = 0
message_interval = 5
counter = 0

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())
```

Source: Own Creation.



MQTT Client Quality Control



main.py 9

Final_Project_sources > RoboticarmSources > main.py > ...

```python
from machine import Pin, PWM
import time

# Servo setup
servo_pin1 = Pin(2)  # D4
servo_pin2 = Pin(4)  # D2
servo_pin3 = Pin(5)  # D1

servo1 = PWM(servo_pin1, freq=50)
servo2 = PWM(servo_pin2, freq=50)
servo3 = PWM(servo_pin3, freq=50)

# Global flag to control servo movement
continue_movement = False

def set_angle(servo, angle):
    duty = int(((angle / 180) * 115) + 40)
    servo.duty(duty)

def move_servos():
    global continue_movement
    continue_movement = True
    for i in range(1):
        if not continue_movement:
            break

        # Servo 1
        set_angle(servo1, 0)
        print("Servo 1 moved to 0 degrees")
        time.sleep(1)
        if not continue_movement:
            break
        set_angle(servo1, 90)
        print("Servo 1 moved to 90 degrees")
        time.sleep(1)
        if not continue_movement:
            break
```

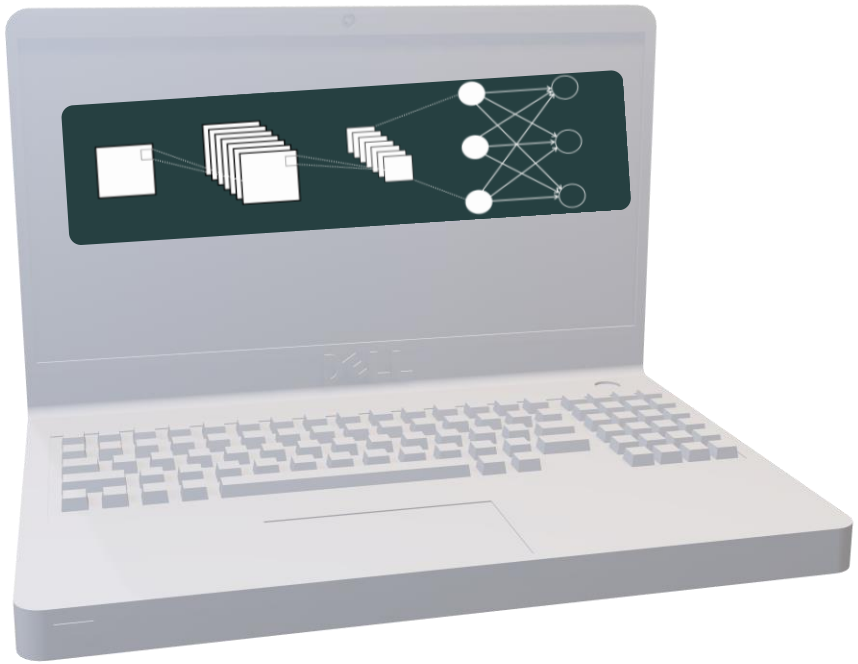
main.py 9

Final_Project_sources > RoboticarmSources > main.py > ...

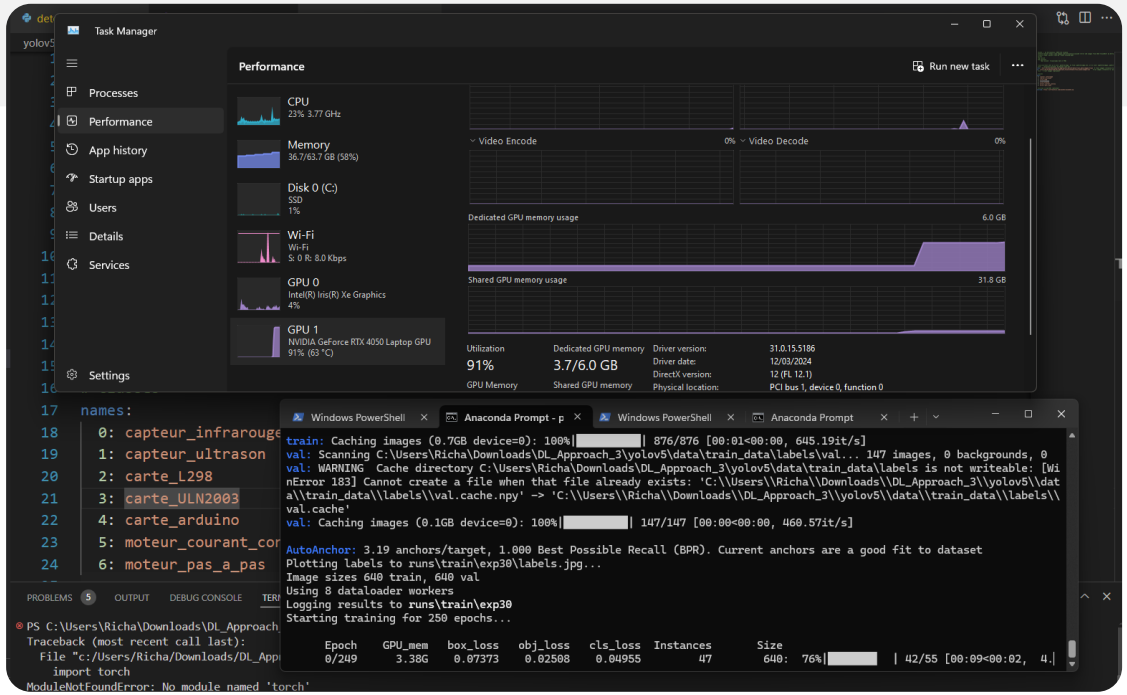
```python
def sub_cb(topic, msg):
            print('Starting servo movement')
            move_servos()
        elif msg == b'1':
            print('Stopping servos')
            stop_servos()

def connect_and_subscribe():
    global client_id, mqtt_server, topic_sub
    client = MQTTClient(client_id, mqtt_server, user=mqtt_user, password=mqtt_pass)
    client.set_callback(sub_cb)
    client.connect()
    client.subscribe(topic_sub)
    print('Connected to %s MQTT broker, subscribed to %s topic' % (mqtt_server, topic_sub))
    return client

def restart_and_reconnect():
    print('Failed to connect to MQTT broker. Reconnecting...')
    time.sleep(10)
    machine.reset()

try:
    client = connect_and_subscribe()
except OSError as e:
    restart_and_reconnect()

while True:
    try:
        client.check_msg()
        if (time.time() - last_message) > message_interval:
            msg = b'Hello #%d' % counter
            client.publish(topic_pub, msg)
            last_message = time.time()
            counter += 1
    except OSError as e:
        restart_and_reconnect()
```

Source: Own Creation.

IIoT – Quality Control - CNN



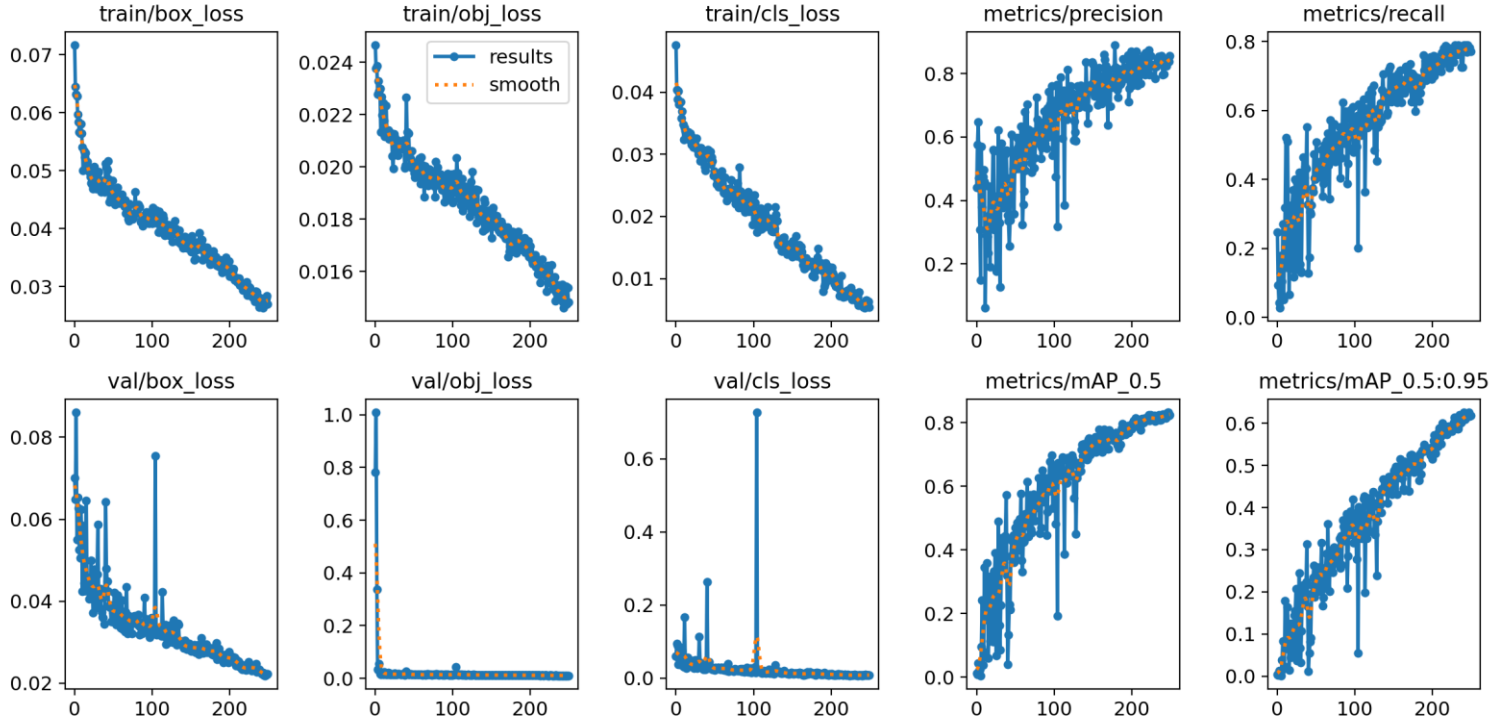
- CNN Algorithm for Object Detection Task: Trained 6 classes for electronic component identification



Source: Own Creation.

IIoT – Quality Control - CNN

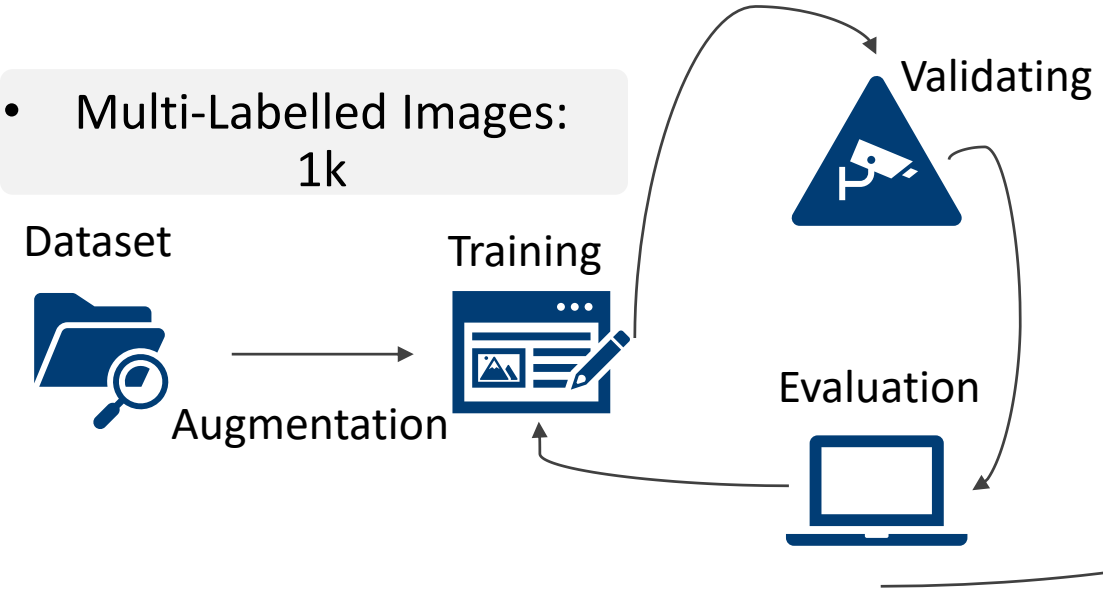
- CNN Algorithm for Object Detection Task: Trained 6 classes for electronic component identification



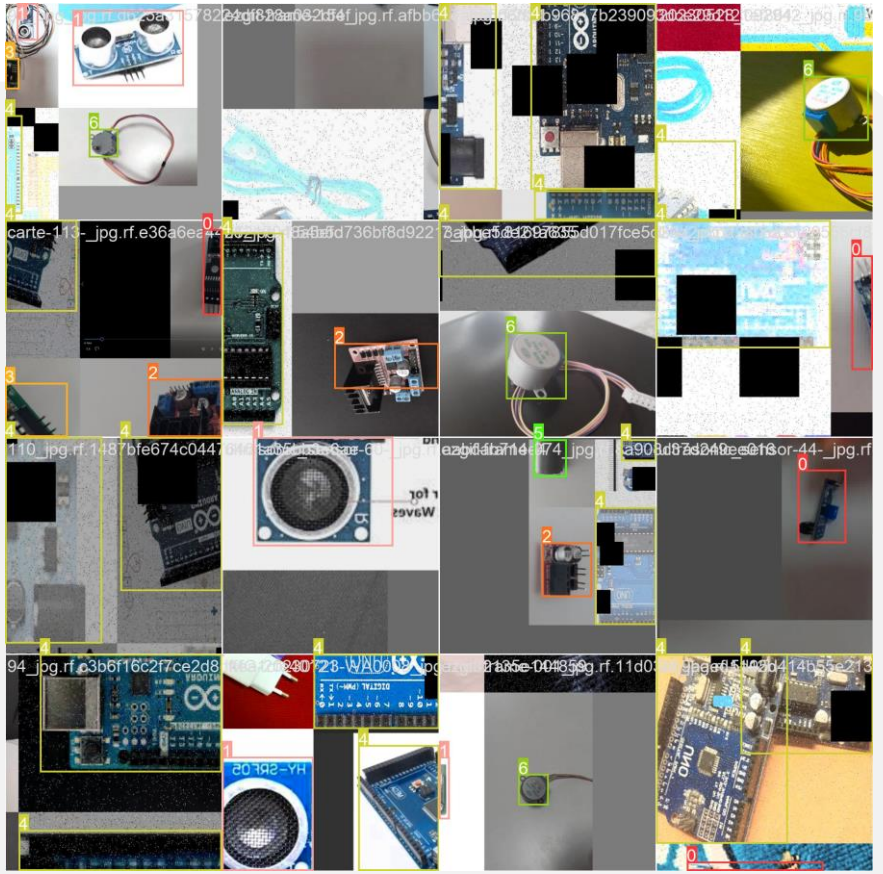
Source: Own Creation.

Object Detection Methodology

- Multi-Labelled Images: 1k



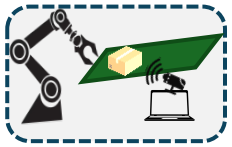
Object Detection Model



Source: Own Creation.

IIoT – Quality Control Main Sources Code

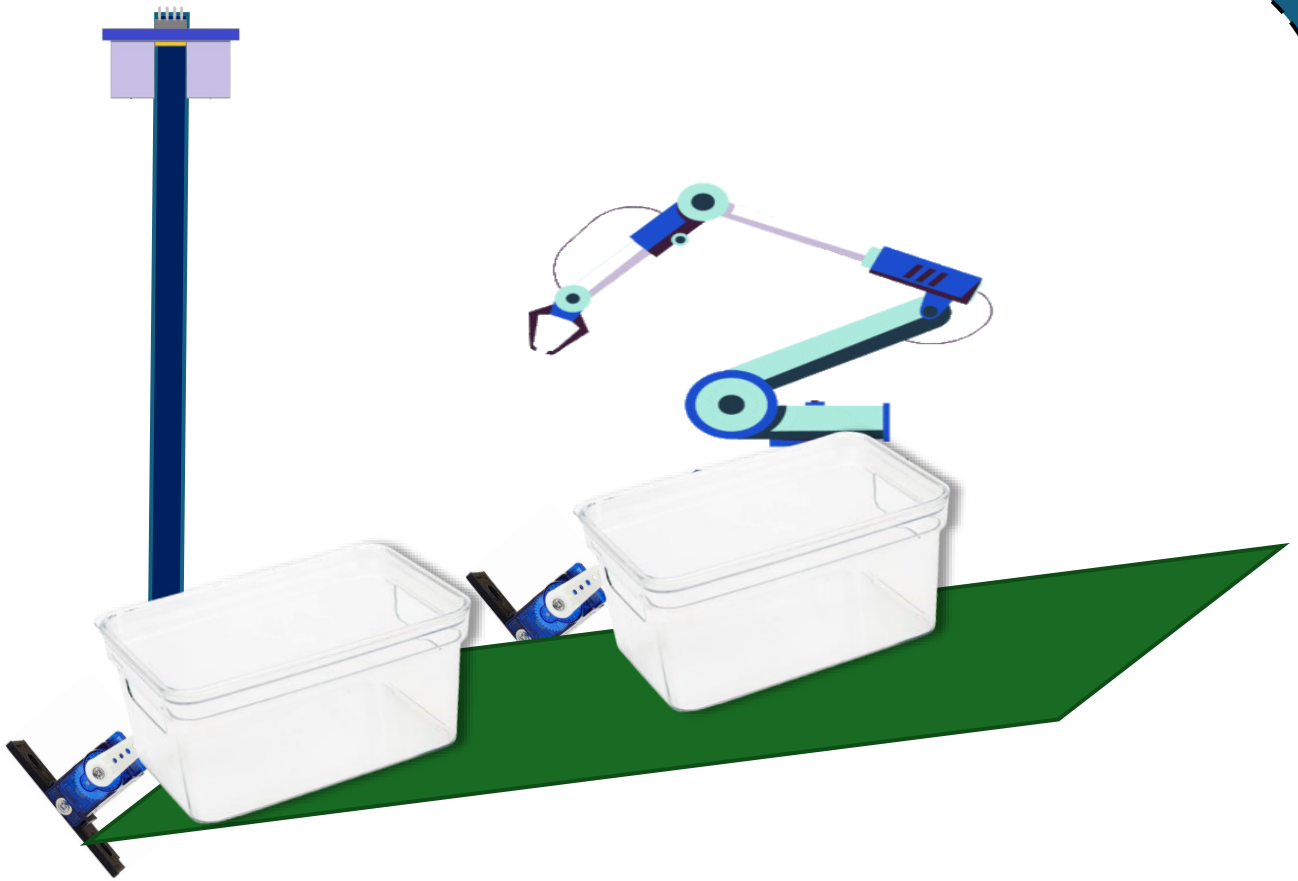

```python
"""

import argparse
import csv
import os
import platform
import sys
from pathlib import Path

import torch
import paho.mqtt.client as mqtt

#  MQTT Variables
MQTT_HOST = "192.168.0.59"
MQTT_PORT = 1883
MQTT_KEEPALIVE_INTERVAL = 5
MQTT_TOPIC = "bin/updatescam"


# on_connect event Handler
def on_connect(mosq, obj, rc):
    print("Connected to MQTT Broker")

# on_publish event Handler
def on_publish(client, userdata, mid):
    print("Message Published...")

# Initiate MQTT Client
mqttc = mqtt.Client()

# Register Event Handlers
mqttc.on_publish = on_publish
mqttc.on_connect = on_connect

# Connect with MQTT Broker
mqttc.connect(MQTT_HOST, MQTT_PORT, MQTT_KEEPALIVE_INTERVAL)

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from ultralytics.utils.plotting import Annotator, colors, save_one_box

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadScreenshots, LoadStreams
from utils.general import (LOGGER, Profile, check_file, check_img_size, check_imshow, check_requirements, colorstr, cv2,
                           increment_path, non_max_suppression, print_args, scale_boxes, strip_optimizer, xyxy2xywh)
from utils.torch_utils import select_device, smart_inference_mode


@smart_inference_mode()
def run(
```



MQTT Client Quality Control

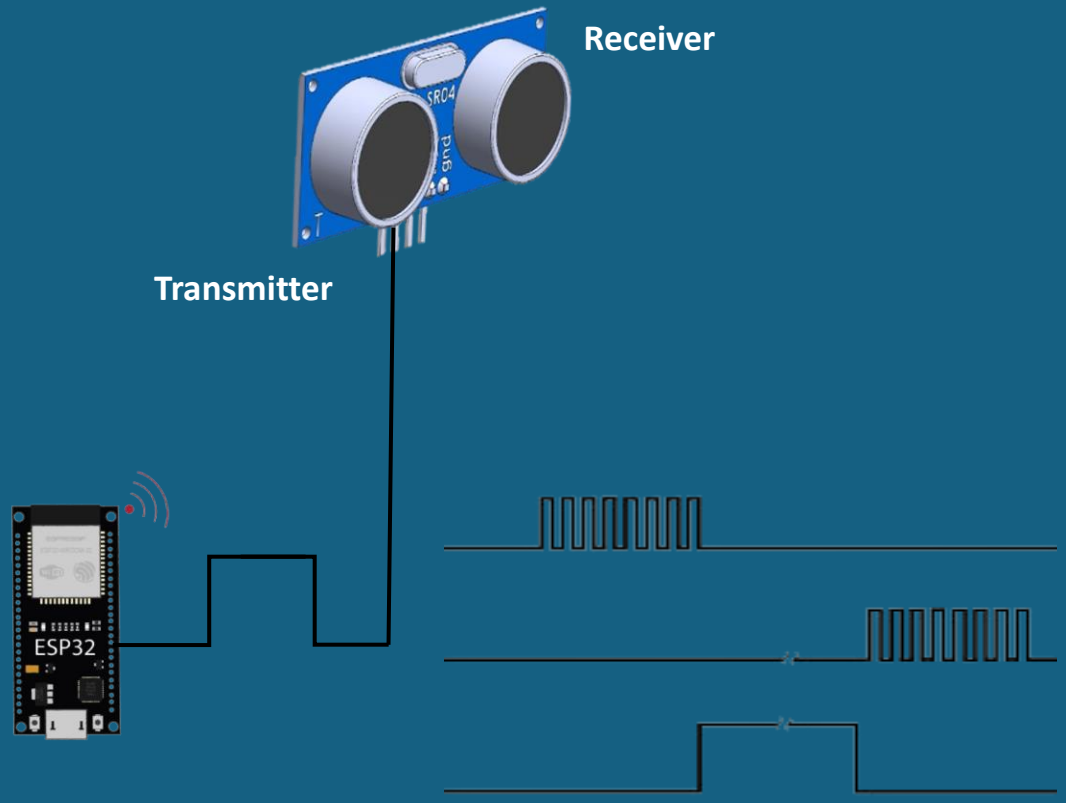
```python
def run(
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'yolov5s.pt', help='model path or triton URL')
    parser.add_argument('--source', type=str, default=ROOT / 'data/images', help='file/dir/URL/glob/screen/0(webcam)')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-csv', action='store_true', help='save results in CSV format')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX inference')
    parser.add_argument('--vid-stride', type=int, default=1, help='video frame-rate stride')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt


def main(opt):
    check_requirements(ROOT / 'requirements.txt', exclude=('tensorboard', 'thop'))
    run(**vars(opt))


if __name__ == '__main__':
    opt = parse_opt()
    main(opt)
```

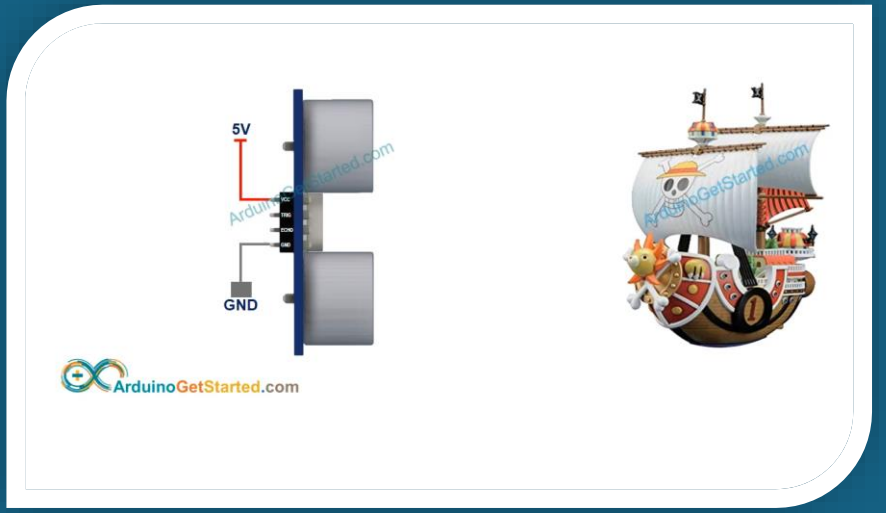
Stock Management with MQTT



Used Components

Ultra Sonic distance Sensor





Echo pulse high time

It is also total travel time

Distance = [340 m/s * Echo pulse high time]/2

Used Components

Servo motors

- First Response
- Precise Control
- Programmable

ESP32

- Smaller Size
- Widely available
- Less cost
- Easy to use

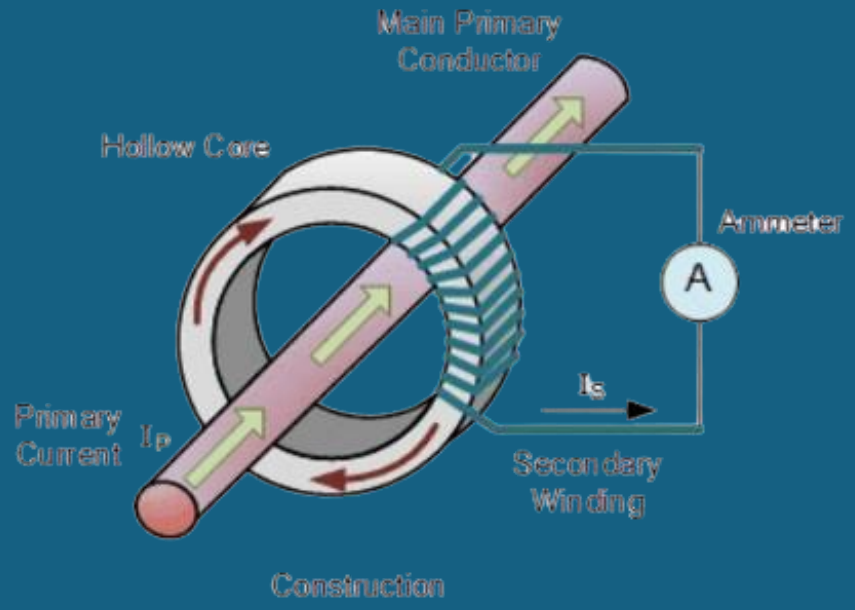
Final implementation of Stock Management

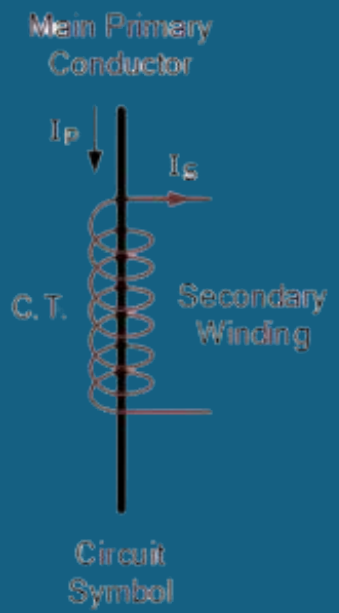
Power Monitoring and controlling system

Used Components

SCT-013 Current Sensor

- 0-30A AC Current
- -25 °C~+70 °C
- Low price

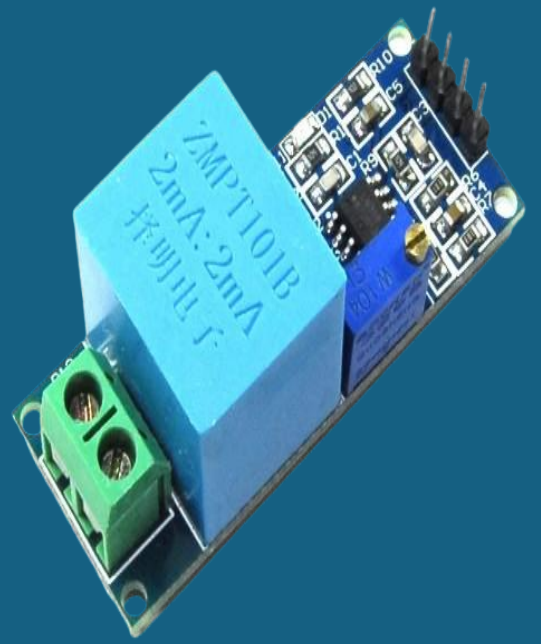




Used Components

ZMPT101B AC Single Phase Voltage Sensor

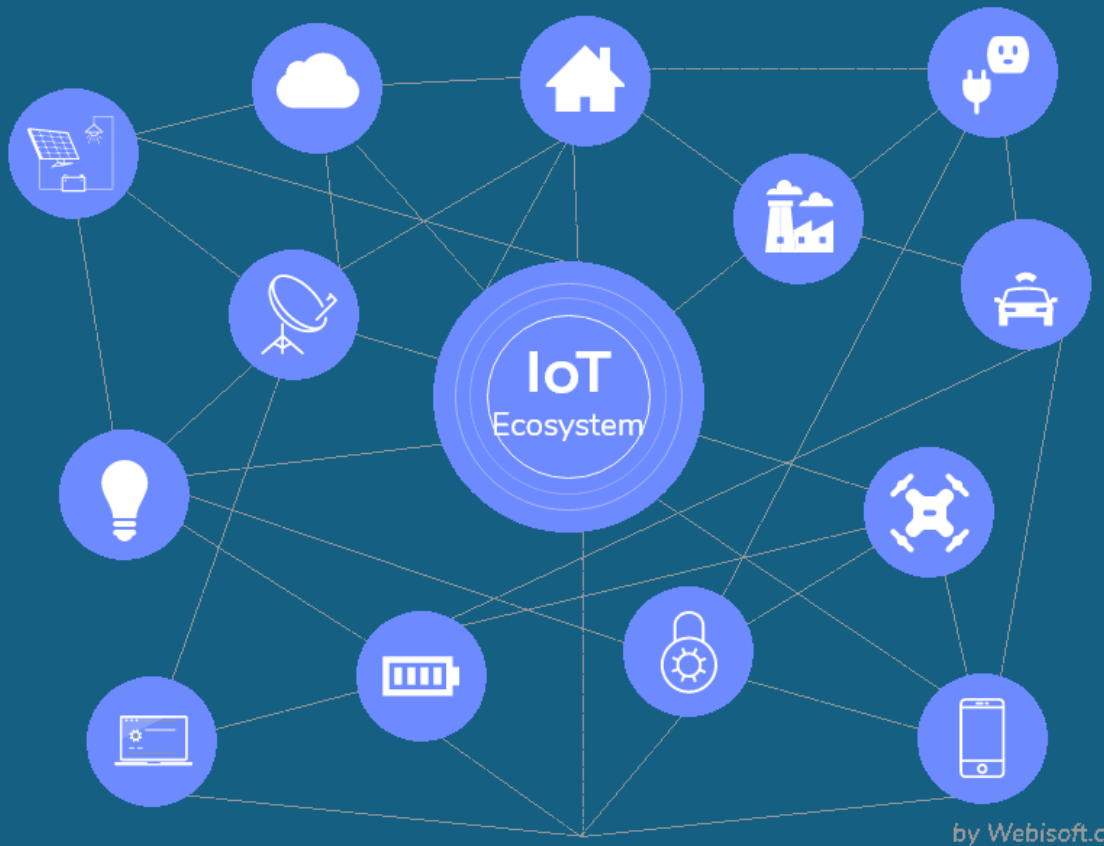
- Voltage up to 250 volts can be measured
- -40 °C~+70 °C
- Supply voltage 5 volts to 30 volts



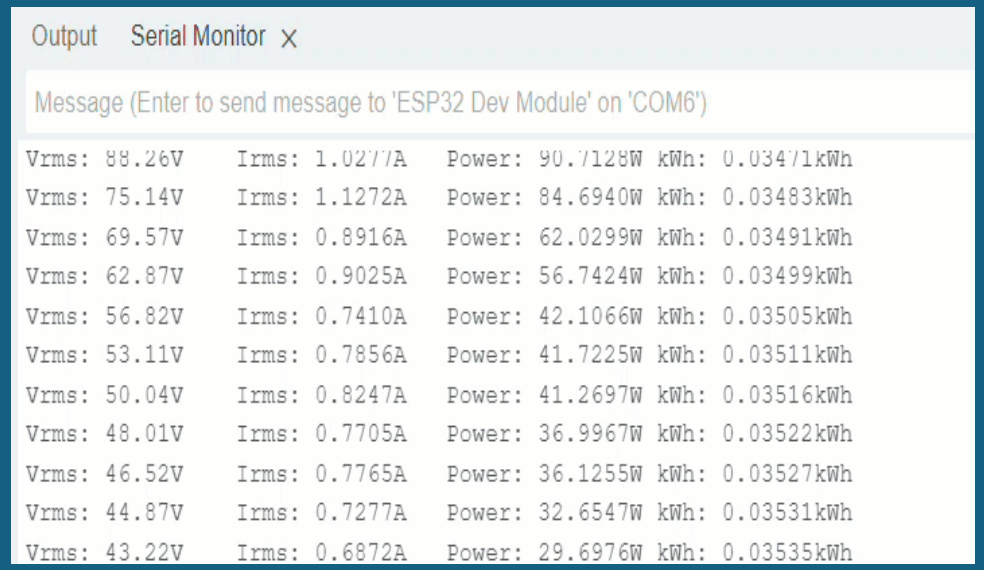
Used Components

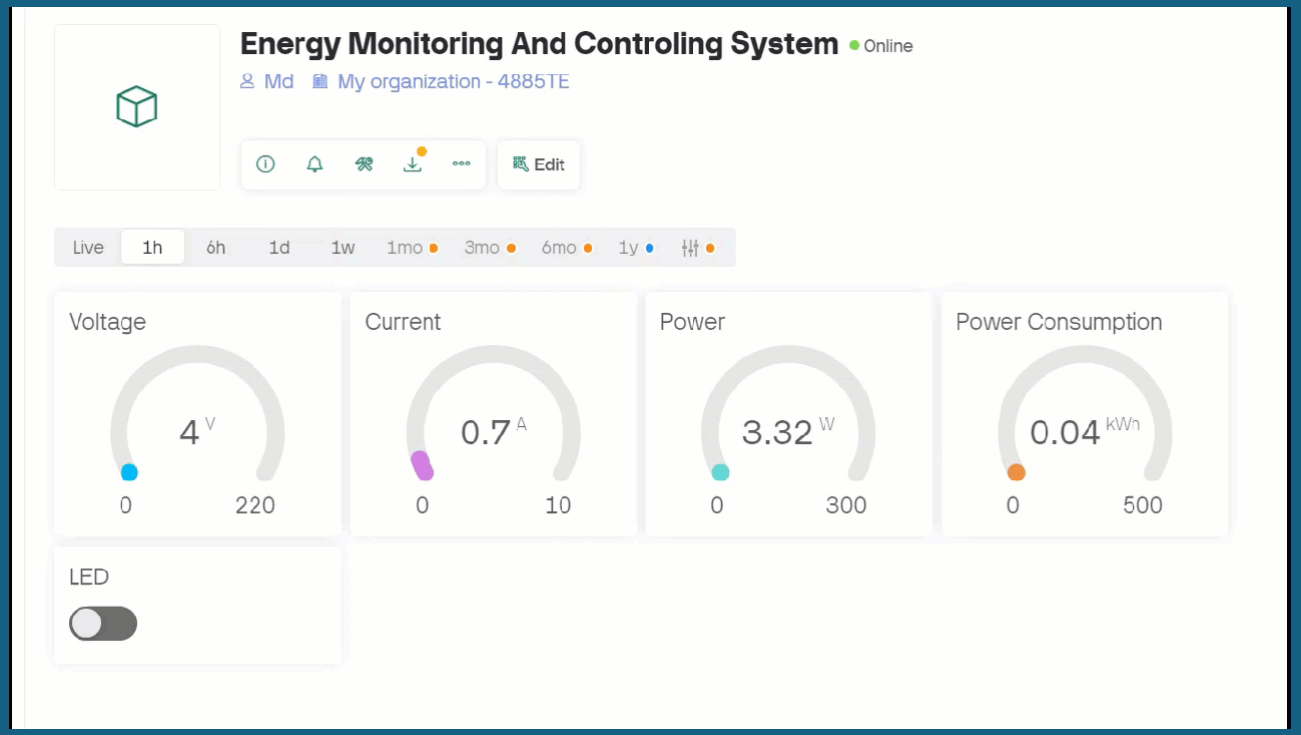
Blynk IOT

- Easy to use
- Publishing and subscribing both possible
- Free for two projects



Final implementation of power monitoring and control system





Safety and security system

Used Components

MQ2 Gas Sensor

- Sensitivity
- Only 5V operating voltage
- Low Cost

Final implementation of power monitoring and control system