TROLL-E

GUIDED BY

Prof.SHABEER S

ROHITH S	B20ECA59
MOHAMMED SANOOF T	B20ECA45
C J ADITHYAN	B20ECA19
JASIM J	B20ECA32

CONTENTS

1.Problem statement

2.Objective

3.Methodology

4.Result and discussion

5.Components Used

6.Future scope

7.Conclusion

8.Reference

PROBLEM STATEMENT

- Billing counters in traditional shopping markets have proven to be a time wasting process for the modern customer.
- People spend hectic time searching for items required.
- In conventional stores, customers typically need to scan manually each item individually at the checkout counter or use self-checkout machines
- Frictionless shopping experience is obtained as customers can purchase their desired product and leave without store staff interaction.

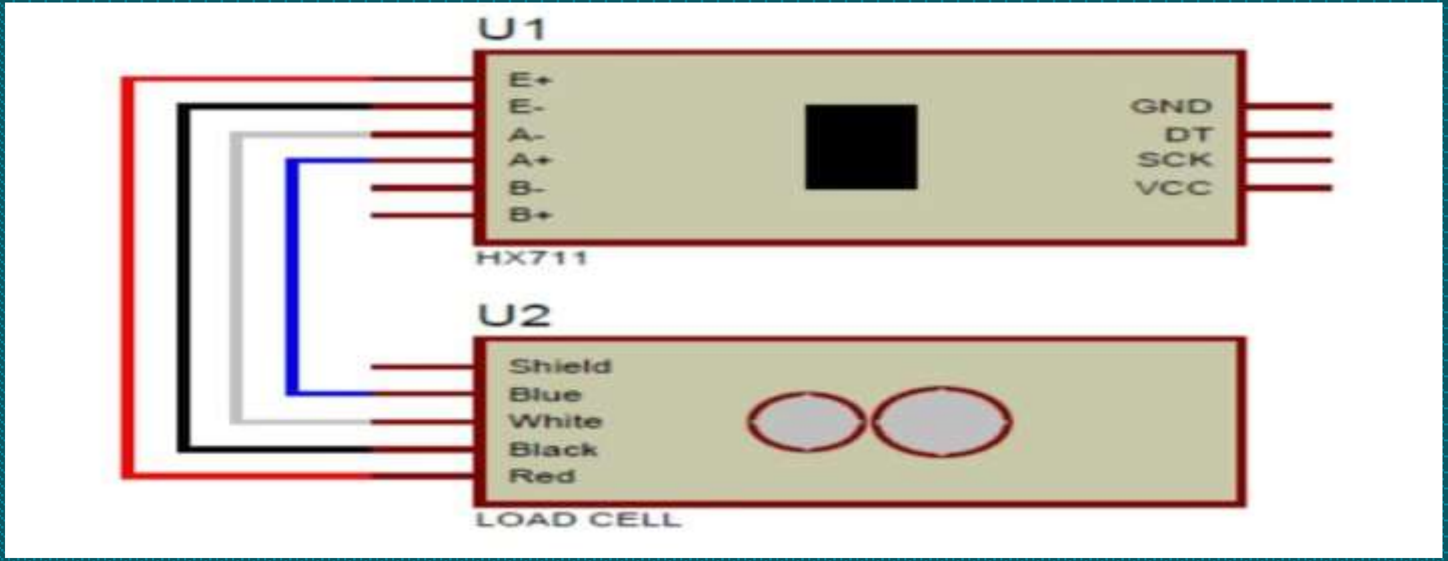
RFID VS ML SHOPPING CART

- RFID scanners require products to be equipped with RFID tags for identification.
- This can be time-consuming for retailers, as they need to affix tags to every item.
- AI shopping system uses cameras and sensors to detect and track the items customers pick up
- ML shopping system provides real-time updates and instant checkout on the items in a customer's cart.
- RFID scanners are primarily designed for fixed scanning points, such as at the entrance or exit of a store
- Implementing RFID technology requires significant infrastructure investment, including RFID readers, tags, and associated software systems

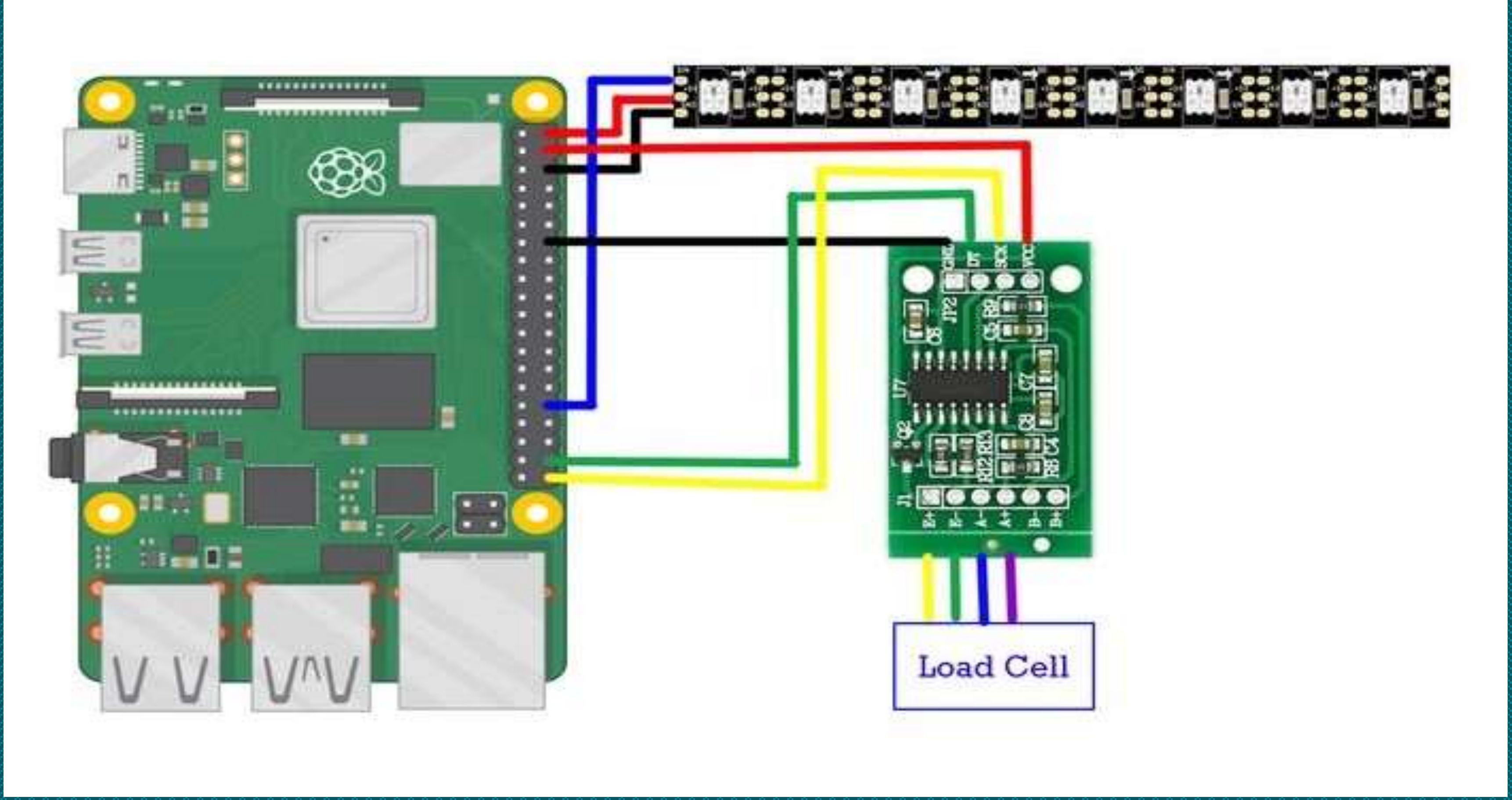
OBJECTIVE

- Main objective of this project is an automated billing system without the use of RFID cards that reduce the manufacturing expense of the product and save the time of the customers.
- The project aims to eliminate the need for traditional checkout counters and cashier interactions.
- Seamless and convenient shopping experience, customers can simply place items in their cart and leave the store without billing counters
- Real-time cart updates and instant payment

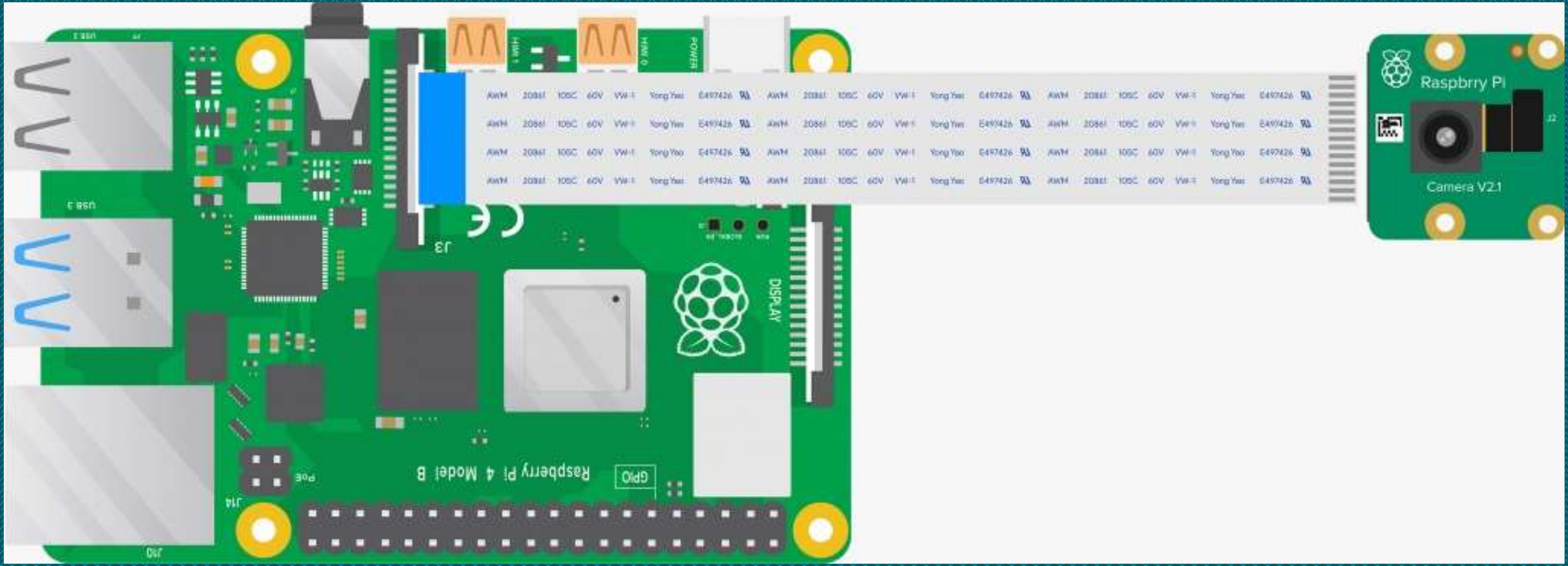
METHODOLOGY



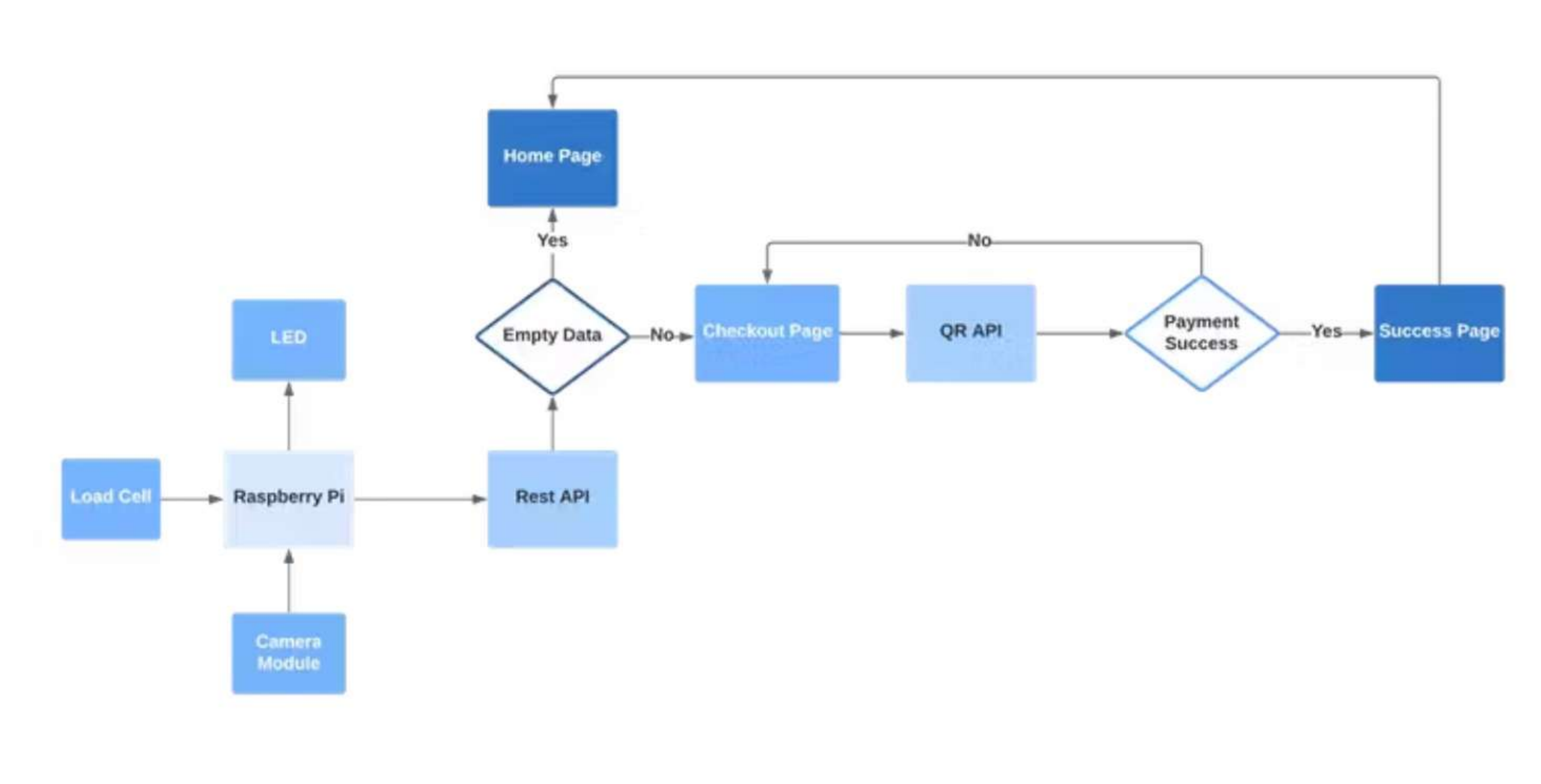
Interfacing of load cell and HX711



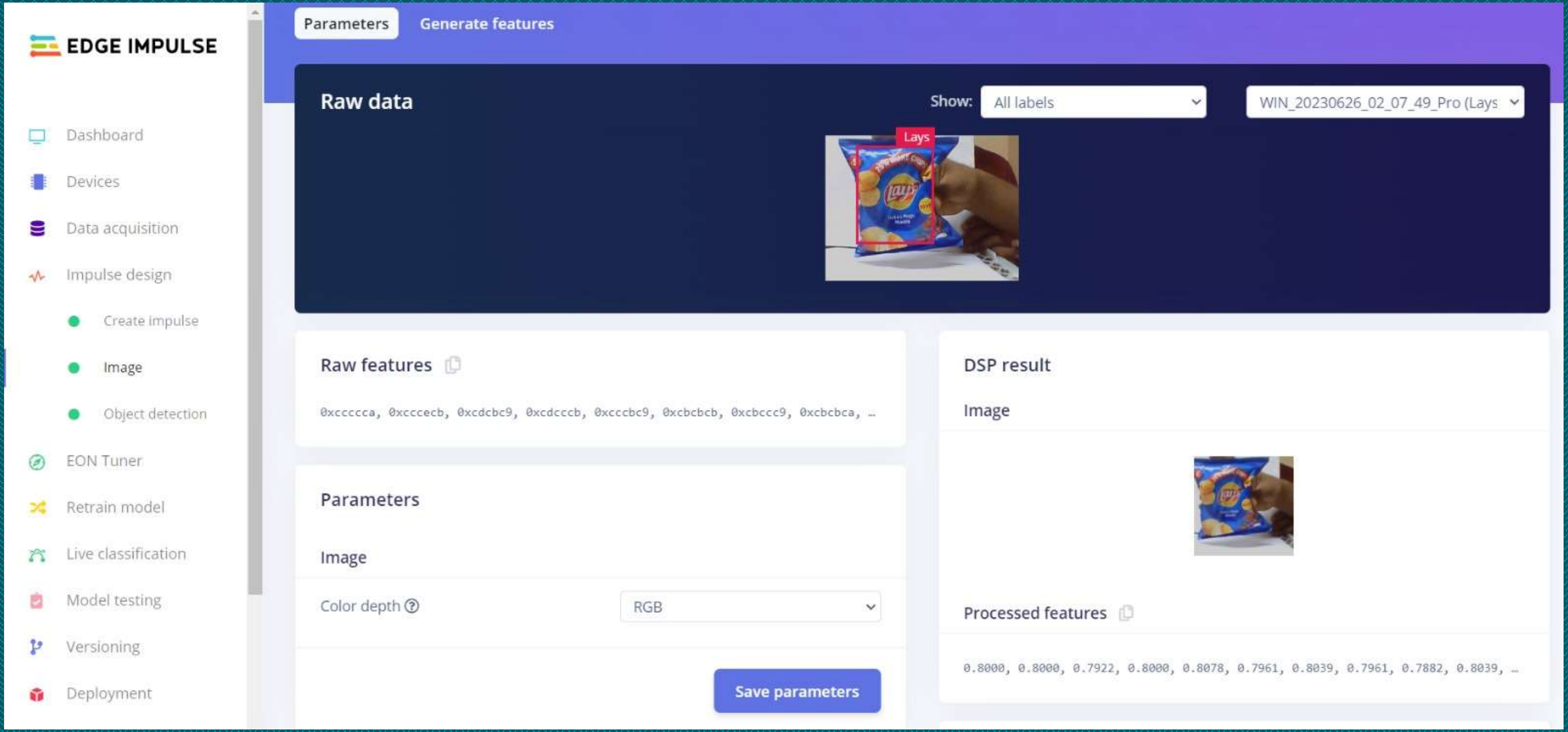
Circuit Diagram



Interfacing of Raspberry pi with camera module



Flow Diagram



Raw data

Show: All labels

WIN_20230626_02_07_49_Pro (Lays

Raw features

```
0xccccca, 0xcccecb, 0xcdcbc9, 0xcdcccb, 0xcccbc9, 0xcbcbcb, 0xcbccc9, 0xcbcbca, …
```

Parameters

Image

Color depth: RGB

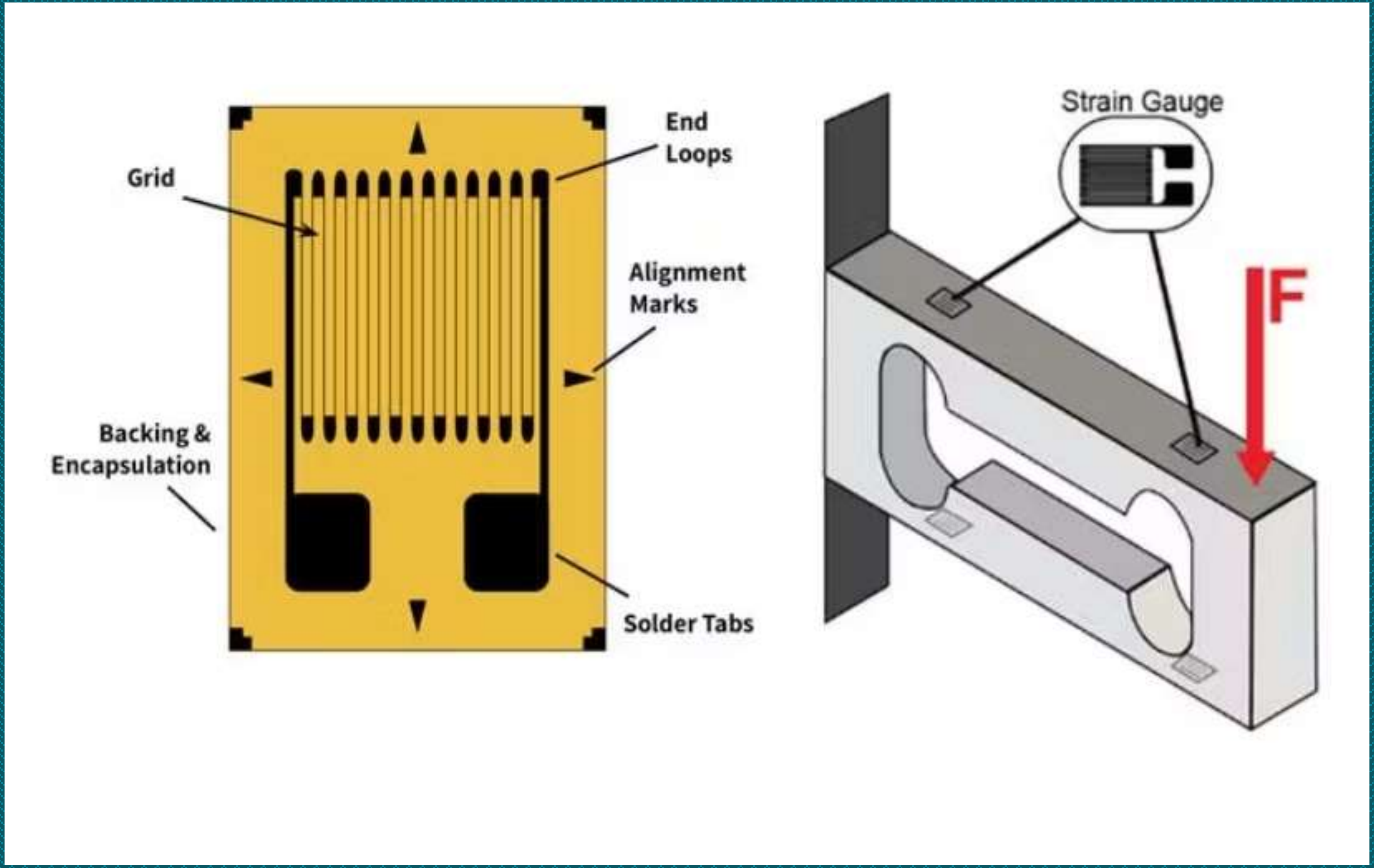
DSP result

Image

Processed features

```
0.8000, 0.8000, 0.7922, 0.8000, 0.8078, 0.7961, 0.8039, 0.7961, 0.7882, 0.8039, …
```

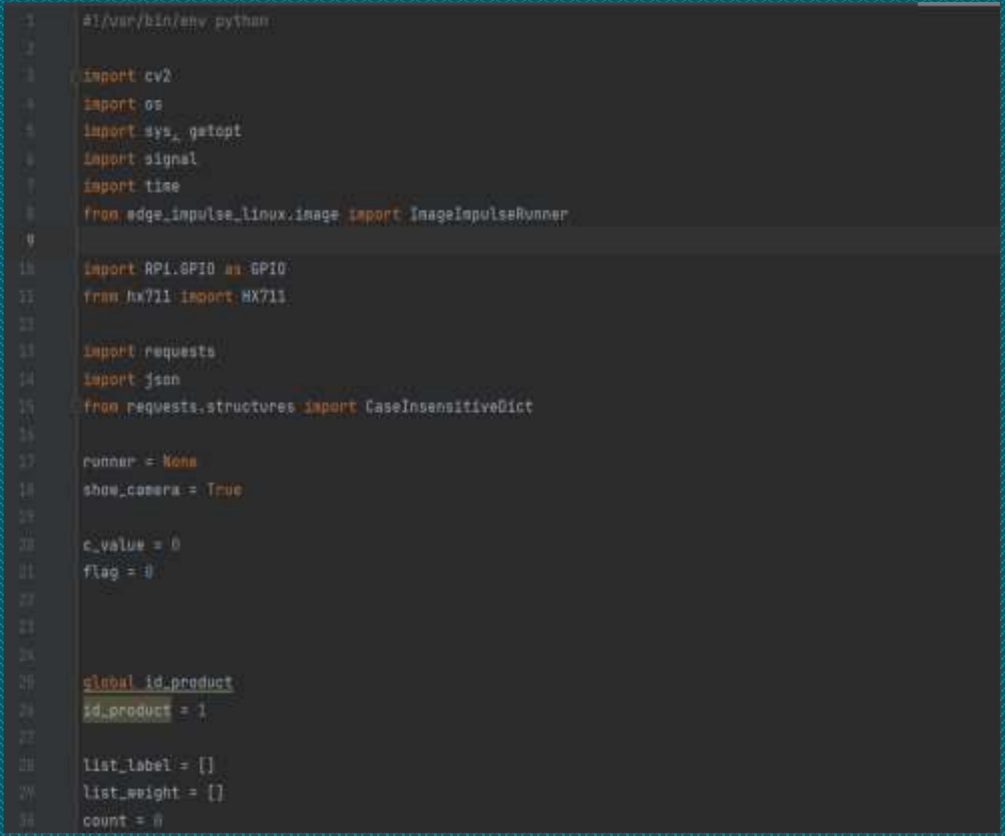
Training Data Set on Edge Impulse


End Loops
Grid
Alignment Marks
Backing & Encapsulation
Solder Tabs
Strain Gauge
F

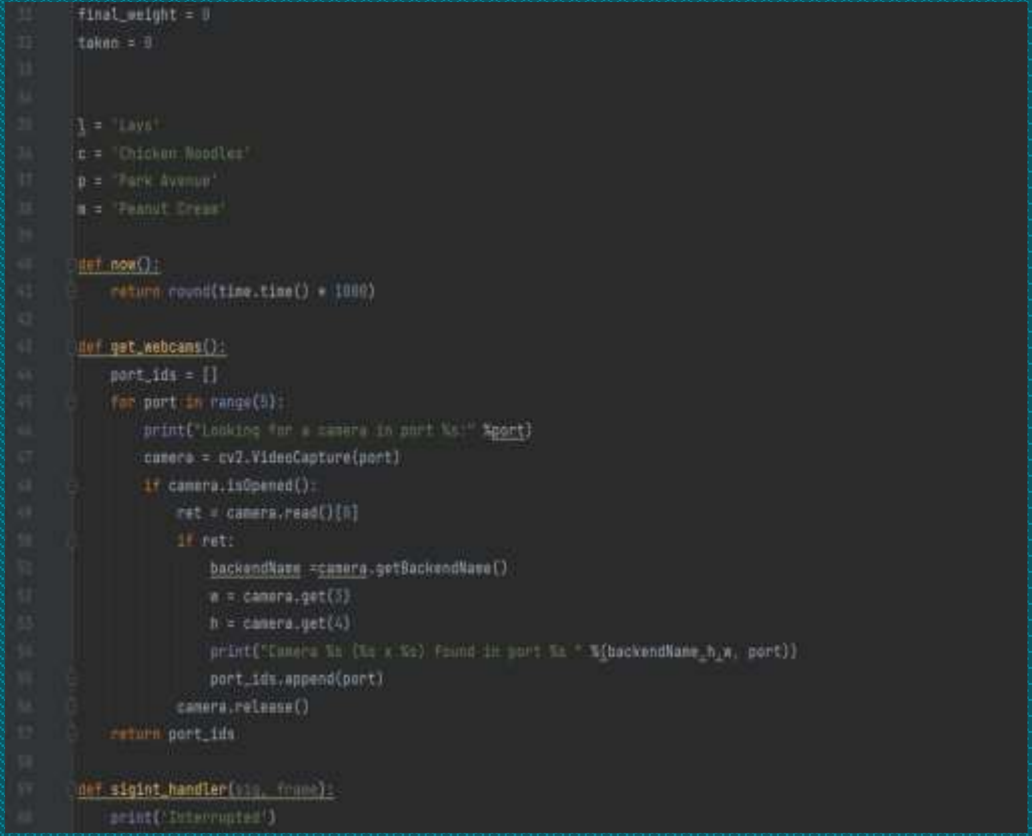
Load cell

CODES

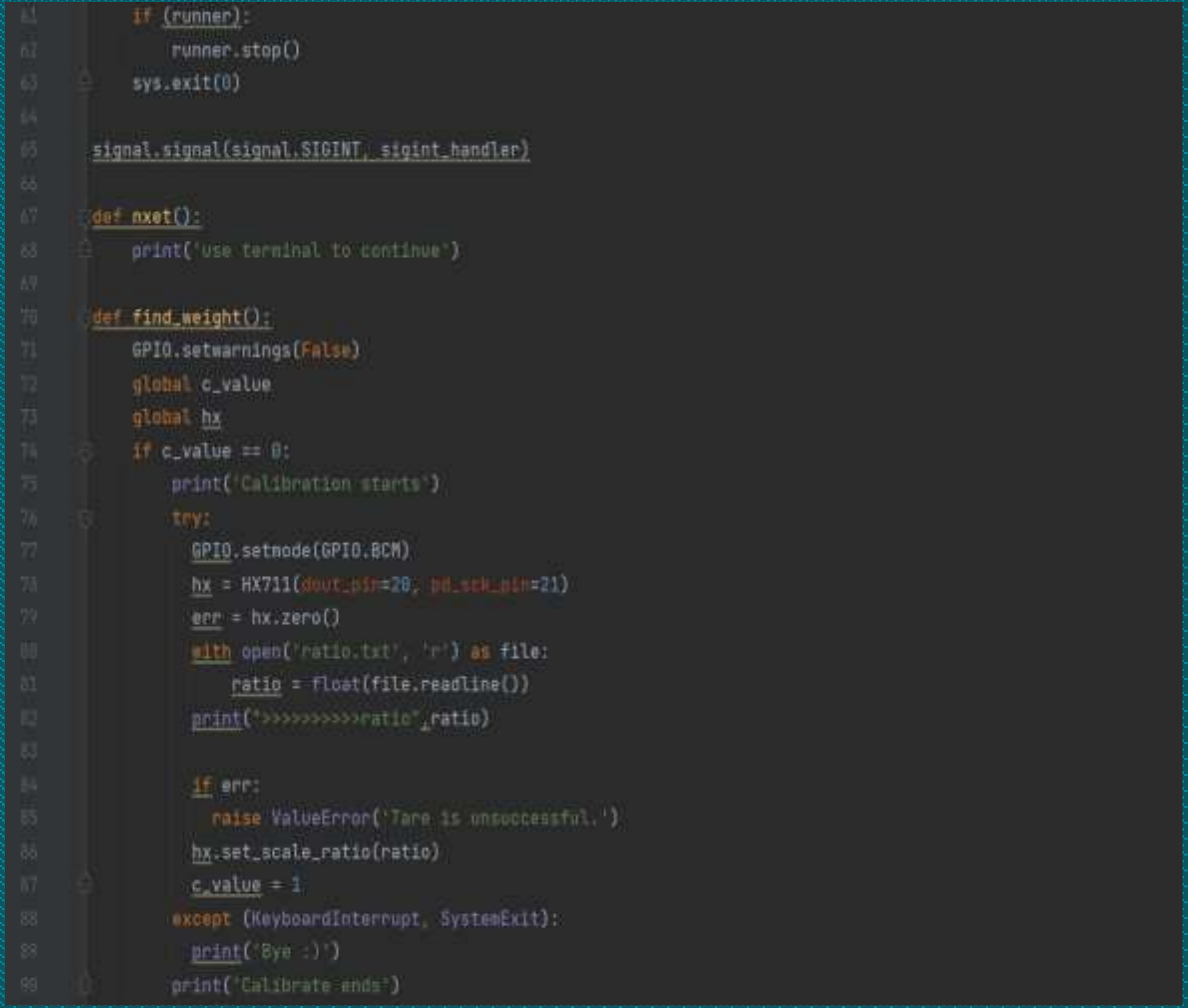
Billing section



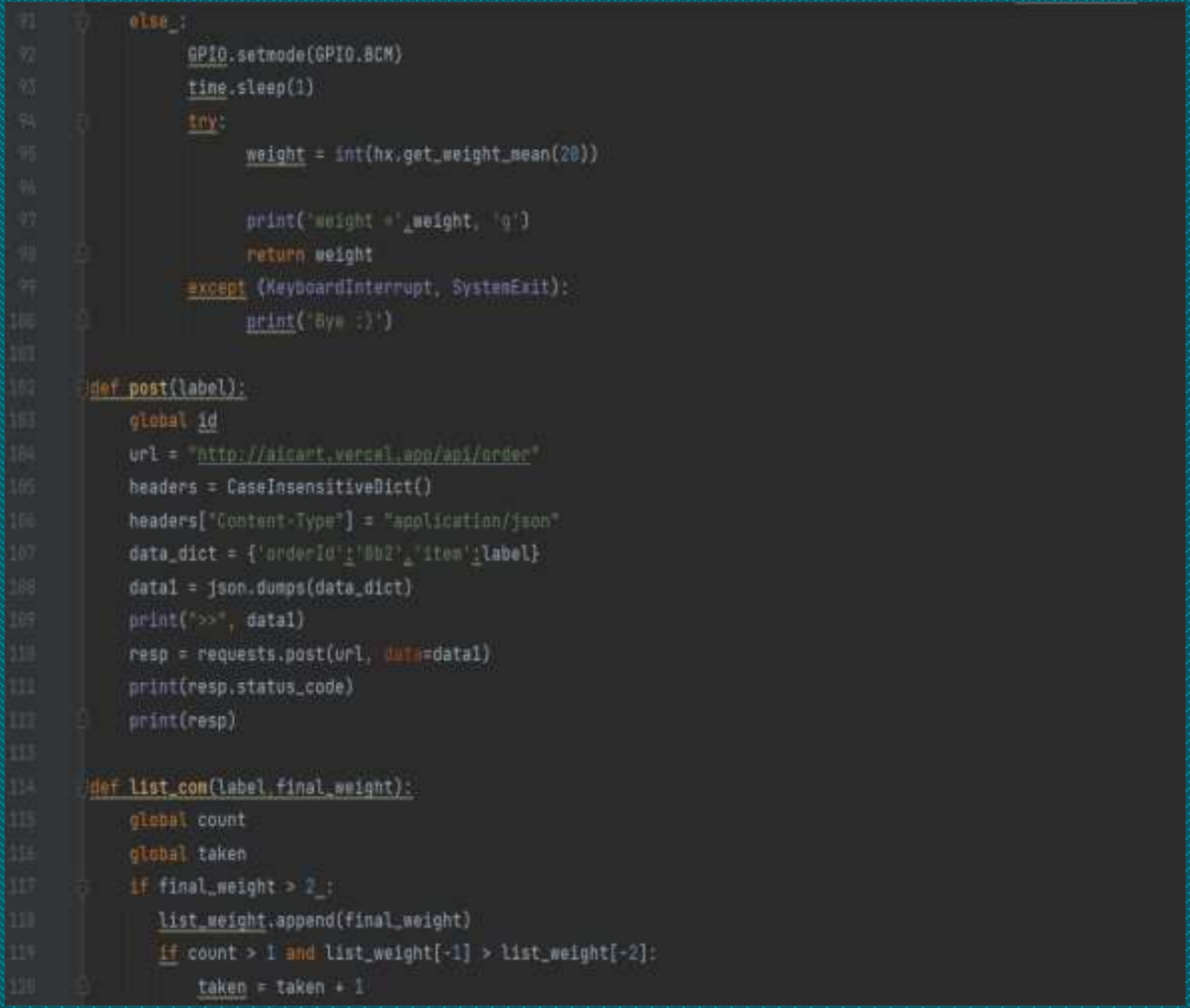
```python
#!/usr/bin/env python

import cv2
import os
import sys, getopt
import signal
import time
from edge_impulse_linux.image import ImageImpulseRunner

import RPi.GPIO as GPIO
from hx711 import HX711

import requests
import json
from requests.structures import CaseInsensitiveDict

runner = None
show_camera = True

c_value = 0
flag = 0



global id_product
id_product = 1

list_label = []
list_weight = []
count = 0
```



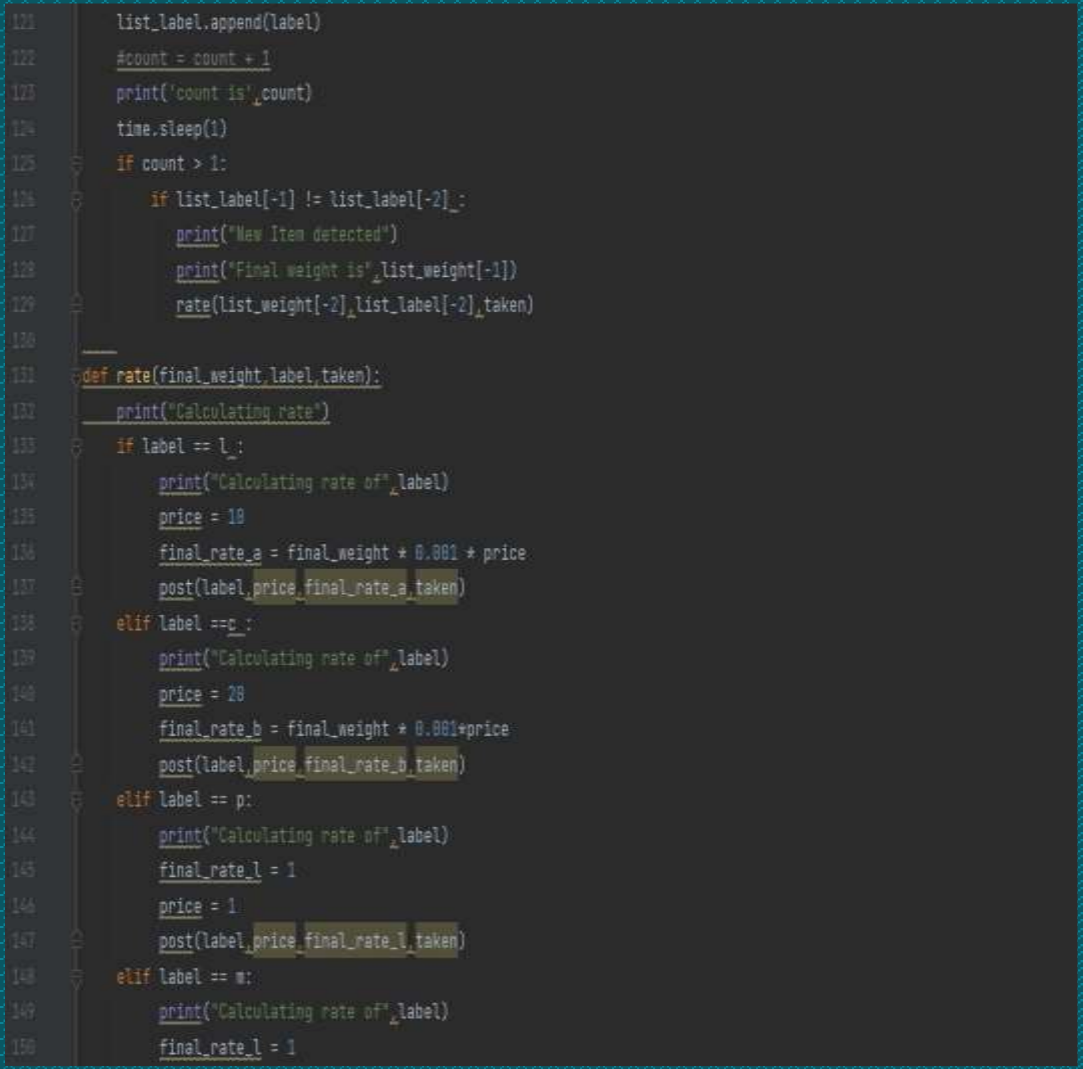
```python
final_weight = 0
taken = 0


l = 'Lays'
c = 'Chicken Noodles'
p = 'Park Avenue'
m = 'Peanut Cream'

def now():
    return round(time.time() * 1000)

def get_webcams():
    port_ids = []
    for port in range(5):
        print("Looking for a camera in port %s:" %port)
        camera = cv2.VideoCapture(port)
        if camera.isOpened():
            ret = camera.read()[0]
            if ret:
                backendName =camera.getBackendName()
                w = camera.get(3)
                h = camera.get(4)
                print("Camera %s (%s x %s) found in port %s " %(backendName,h,w, port))
                port_ids.append(port)
            camera.release()
    return port_ids

def sigint_handler(sig, frame):
    print('Interrupted')
```



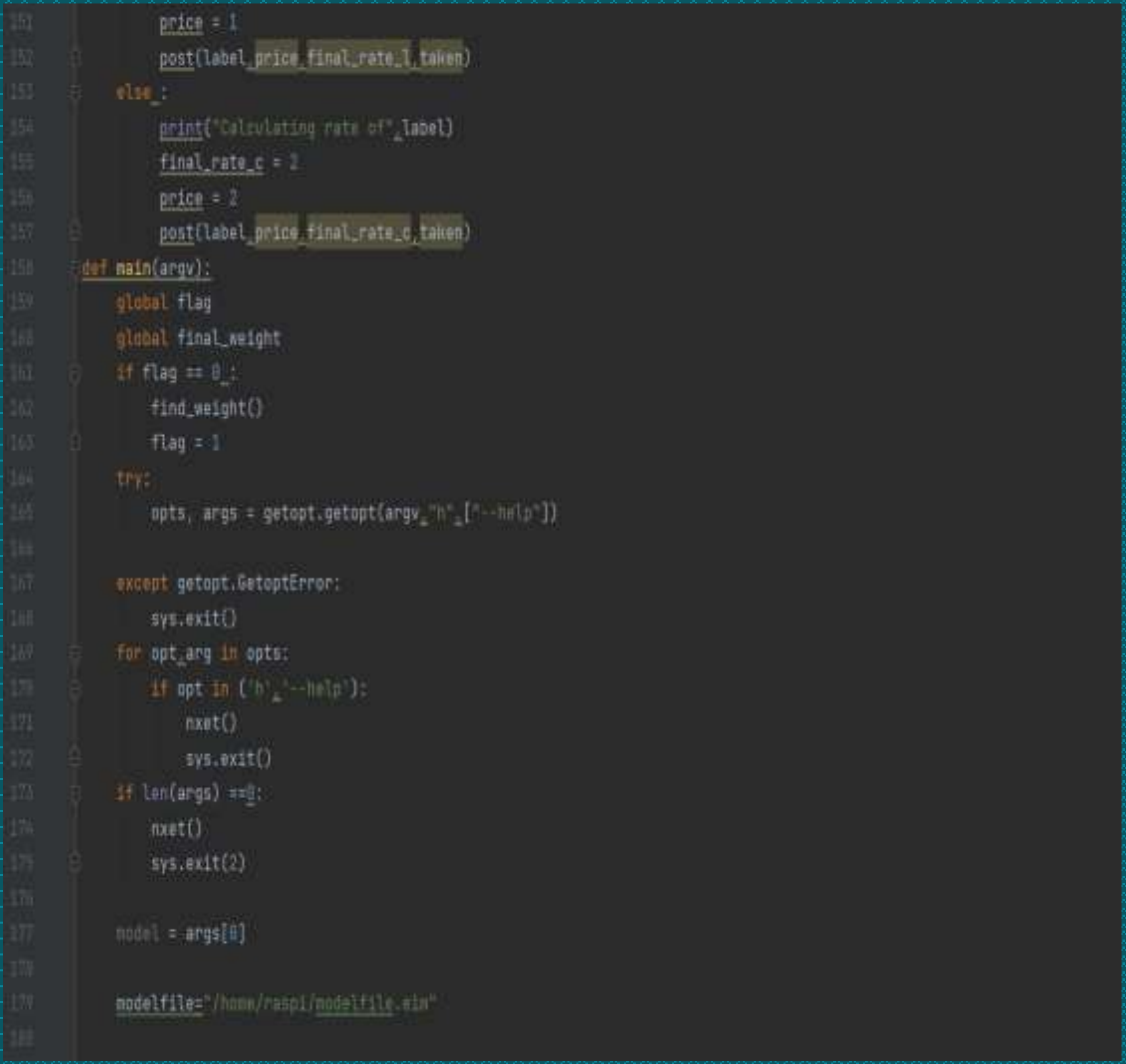
```
    if (runner):
        runner.stop()
    sys.exit(0)

signal.signal(signal.SIGINT, sigint_handler)

def nxet():
    print('use terminal to continue')

def find_weight():
    GPIO.setwarnings(False)
    global c_value
    global hx
    if c_value == 0:
        print('Calibration starts')
        try:
          GPIO.setmode(GPIO.BCM)
          hx = HX711(dout_pin=20, pd_sck_pin=21)
          err = hx.zero()
          with open('ratio.txt', 'r') as file:
              ratio = float(file.readline())
          print(">>>>>>>>>>>ratio",ratio)

          if err:
            raise ValueError('Tare is unsuccessful.')
          hx.set_scale_ratio(ratio)
          c_value = 1
        except (KeyboardInterrupt, SystemExit):
          print('Bye :)')
        print('Calibrate ends')
```



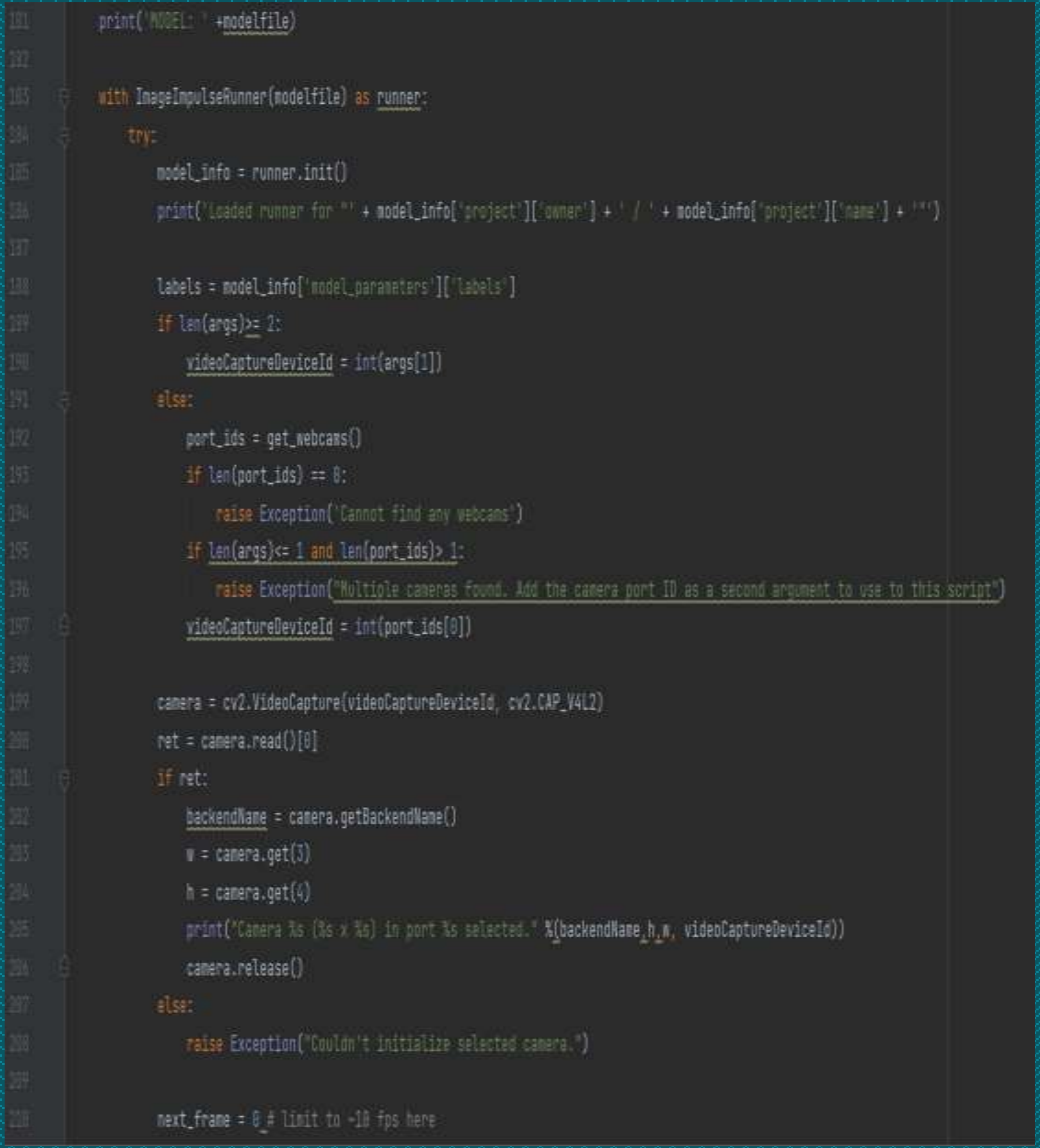
```
    else :
          GPIO.setmode(GPIO.BCM)
          time.sleep(1)
          try:
                weight = int(hx.get_weight_mean(20))

                print('weight =',weight, 'g')
                return weight
          except (KeyboardInterrupt, SystemExit):
                print('Bye :)')

def post(label):
    global id
    url = "http://aicart.vercel.app/api/order"
    headers = CaseInsensitiveDict()
    headers["Content-Type"] = "application/json"
    data_dict = {'orderId':'8b2','item':label}
    data1 = json.dumps(data_dict)
    print(">>", data1)
    resp = requests.post(url, data=data1)
    print(resp.status_code)
    print(resp)

def list_com(label,final_weight):
    global count
    global taken
    if final_weight > 2 :
       list_weight.append(final_weight)
       if count > 1 and list_weight[-1] > list_weight[-2]:
           taken = taken + 1
```



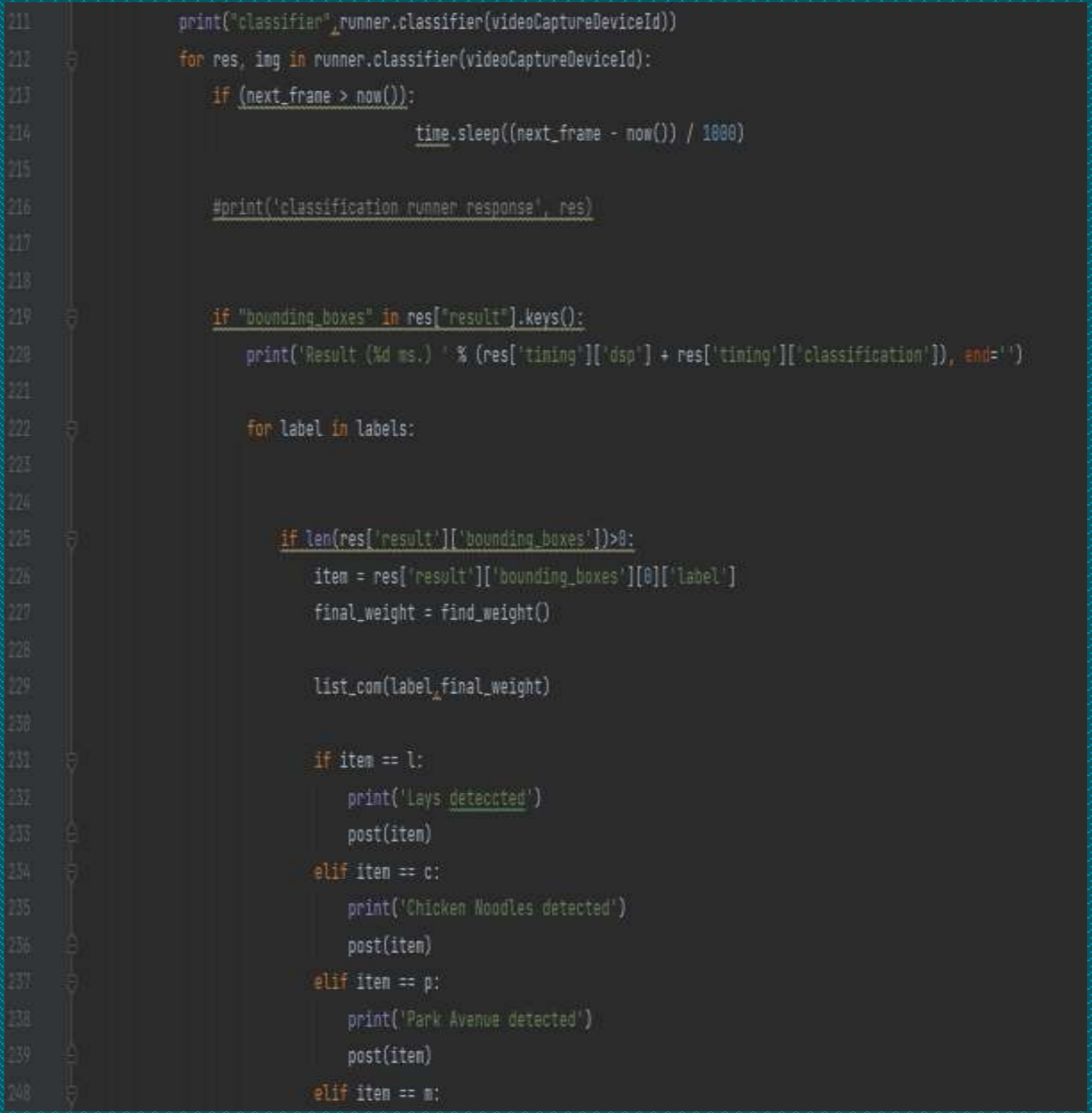
```
    list_label.append(label)
    #count = count + 1
    print('count is',count)
    time.sleep(1)
    if count > 1:
        if list_label[-1] != list_label[-2] :
            print("New Item detected")
            print("Final weight is",list_weight[-1])
            rate(list_weight[-2],list_label[-2],taken)

def rate(final_weight,label,taken):
    print("Calculating rate")
    if label == l :
        print("Calculating rate of",label)
        price = 10
        final_rate_a = final_weight * 0.001 * price
        post(label,price,final_rate_a,taken)
    elif label ==c :
        print("Calculating rate of",label)
        price = 20
        final_rate_b = final_weight * 0.001*price
        post(label,price,final_rate_b,taken)
    elif label == p:
        print("Calculating rate of",label)
        final_rate_l = 1
        price = 1
        post(label,price,final_rate_l,taken)
    elif label == m:
        print("Calculating rate of",label)
        final_rate_l = 1
```



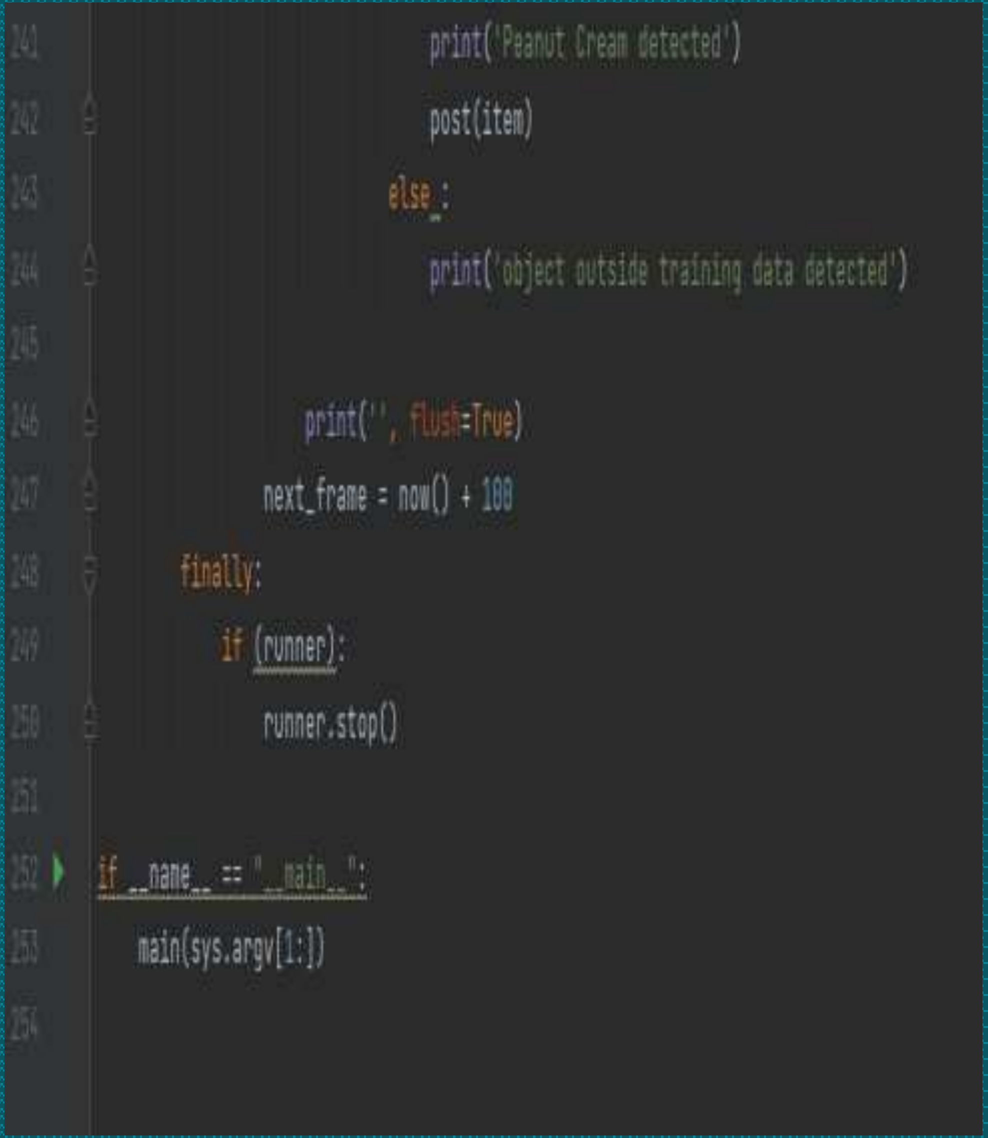
```
        price = 1
        post(label,price,final_rate_l,taken)
    else :
        print("Calculating rate of",label)
        final_rate_c = 2
        price = 2
        post(label,price,final_rate_c,taken)
def main(argv):
    global flag
    global final_weight
    if flag == 0 :
        find_weight()
        flag = 1
    try:
        opts, args = getopt.getopt(argv,"h",["--help"])

    except getopt.GetoptError:
        sys.exit()
    for opt,arg in opts:
        if opt in ('h','--help'):
            nxet()
            sys.exit()
    if len(args) ==0:
        nxet()
        sys.exit(2)

    model = args[0]

    modelfile="/home/raspi/modelfile.eim"
```



```
print('MODEL: ' +modelfile)

with ImageImpulseRunner(modelfile) as runner:
    try:
        model_info = runner.init()
        print('Loaded runner for "' + model_info['project']['owner'] + ' / ' + model_info['project']['name'] + '"')

        labels = model_info['model_parameters']['labels']
        if len(args)>= 2:
            videoCaptureDeviceId = int(args[1])
        else:
            port_ids = get_webcams()
            if len(port_ids) == 0:
                raise Exception('Cannot find any webcams')
            if len(args)<= 1 and len(port_ids)> 1:
                raise Exception("Multiple cameras found. Add the camera port ID as a second argument to use to this script")
            videoCaptureDeviceId = int(port_ids[0])

        camera = cv2.VideoCapture(videoCaptureDeviceId, cv2.CAP_V4L2)
        ret = camera.read()[0]
        if ret:
            backendName = camera.getBackendName()
            w = camera.get(3)
            h = camera.get(4)
            print("Camera %s (%s x %s) in port %s selected." %(backendName,h,w, videoCaptureDeviceId))
            camera.release()
        else:
            raise Exception("Couldn't initialize selected camera.")

        next_frame = 0 # limit to ~10 fps here
```

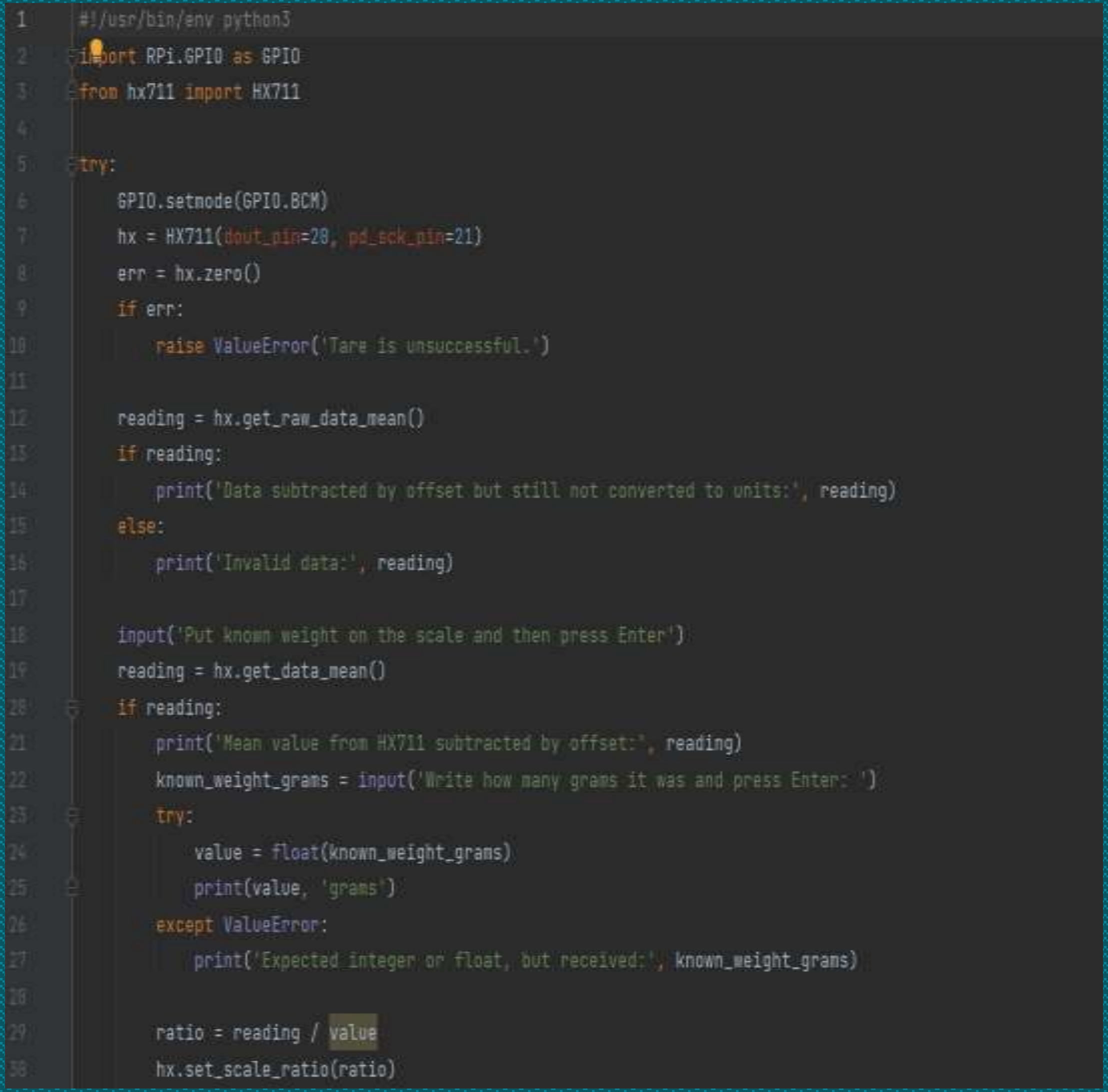


```
            print("classifier",runner.classifier(videoCaptureDeviceId))
            for res, img in runner.classifier(videoCaptureDeviceId):
                if (next_frame > now()):
                                            time.sleep((next_frame - now()) / 1000)

                #print('classification runner response', res)


                if "bounding_boxes" in res["result"].keys():
                    print('Result (%d ms.) ' % (res['timing']['dsp'] + res['timing']['classification']), end='')

                    for label in labels:


                        if len(res['result']['bounding_boxes'])>0:
                            item = res['result']['bounding_boxes'][0]['label']
                            final_weight = find_weight()

                            list_com(label,final_weight)

                            if item == l:
                                print('Lays deteccted')
                                post(item)
                            elif item == c:
                                print('Chicken Noodles detected')
                                post(item)
                            elif item == p:
                                print('Park Avenue detected')
                                post(item)
                            elif item == m:
```

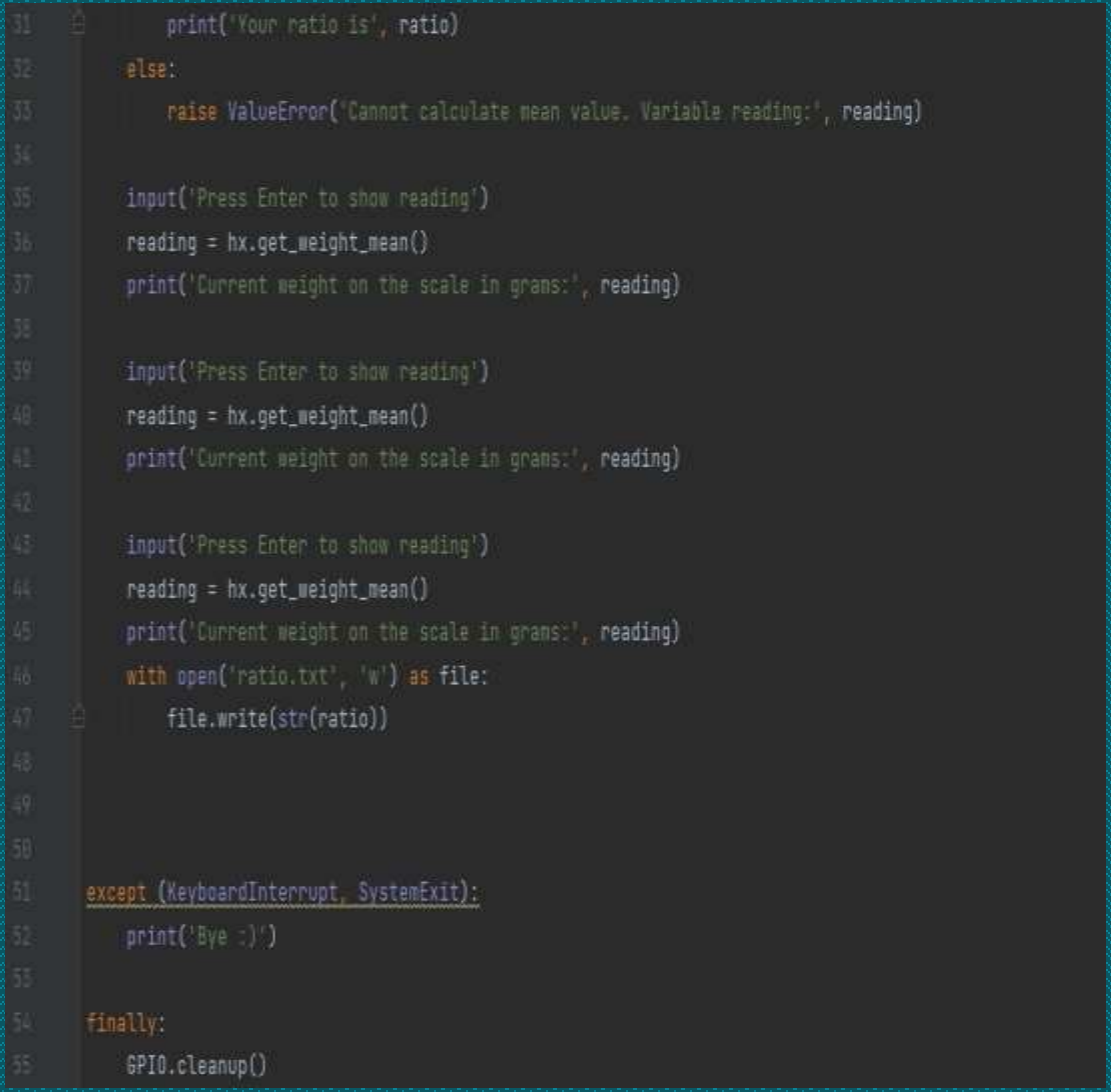


```
                                print('Peanut Cream detected')
                                post(item)
                            else :
                                print('object outside training data detected')

                    print('', flush=True)
                next_frame = now() + 100
        finally:
            if (runner):
                runner.stop()

if __name__ == "__main__":
    main(sys.argv[1:])
```

Callibration section



```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
from hx711 import HX711

try:
    GPIO.setmode(GPIO.BCM)
    hx = HX711(dout_pin=20, pd_sck_pin=21)
    err = hx.zero()
    if err:
        raise ValueError('Tare is unsuccessful.')

    reading = hx.get_raw_data_mean()
    if reading:
        print('Data subtracted by offset but still not converted to units:', reading)
    else:
        print('Invalid data:', reading)

    input('Put known weight on the scale and then press Enter')
    reading = hx.get_data_mean()
    if reading:
        print('Mean value from HX711 subtracted by offset:', reading)
        known_weight_grams = input('Write how many grams it was and press Enter: ')
        try:
            value = float(known_weight_grams)
            print(value, 'grams')
        except ValueError:
            print('Expected integer or float, but received:', known_weight_grams)

        ratio = reading / value
        hx.set_scale_ratio(ratio)
```



```
        print('Your ratio is', ratio)
    else:
        raise ValueError('Cannot calculate mean value. Variable reading:', reading)

    input('Press Enter to show reading')
    reading = hx.get_weight_mean()
    print('Current weight on the scale in grams:', reading)

    input('Press Enter to show reading')
    reading = hx.get_weight_mean()
    print('Current weight on the scale in grams:', reading)

    input('Press Enter to show reading')
    reading = hx.get_weight_mean()
    print('Current weight on the scale in grams:', reading)
    with open('ratio.txt', 'w') as file:
        file.write(str(ratio))



except (KeyboardInterrupt, SystemExit):
    print('Bye :)')

finally:
    GPIO.cleanup()
```

RESULTS AND DISCUSSION

- We have implemented our cart in which a commodity is detected using object detection. Then using load cell we callibrate the weight of the commodity .Price of the commodity is determined using the preset values .The total price that have to be paid by the customer will be displayed in the user interface.
- So as by our project we have reached our aim to build a cart with automated facilities and features that solve the constraints of the traditional shops.

COMPONENTS USED

- Raspberry pi
- Hx-711
- Load cell
- Camera module

FUTURE SCOPE

- Future Troll-E may incorporate advanced ,fast and efficient technologies.
- We can incorporate indoor navigation system on our cart, in order to determine the coordinates where the products were kept.
- Lora based beacon system enables users to navigate inside the shopping mart and to access the required commodity.

CONCLUSION

- Troll-E is an important advancement in modern shopping trends offering numerous benefits to buyers and sellers.
- Throughout the project ,we examined key features such as automated billing system, weight sensing using load cell and a touch interface which shows a temporary bill of the commodities present in the cart.
- Research indicates that automated shopping carts have positive impact on the valuable time of the customers and reduce the rush on the billing counters.

REFERENCE

- Chengji Liu1,Yufan Tao1,Jiawei Liang1, Kai Li1,Yihang Chen1 “Object Detection Based on YOLO Network” in 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC),https://ieeexplore.ieee.org/document/8851911.

- https://www.hackster.io/coderscafe/autobill-042d29.

- N. Anju Latha, B. Rama Murthy,”Raspberry pi based Weighting Scale using Load Cell” journal published on IJSRST, vol 3.

THANK YOU!