

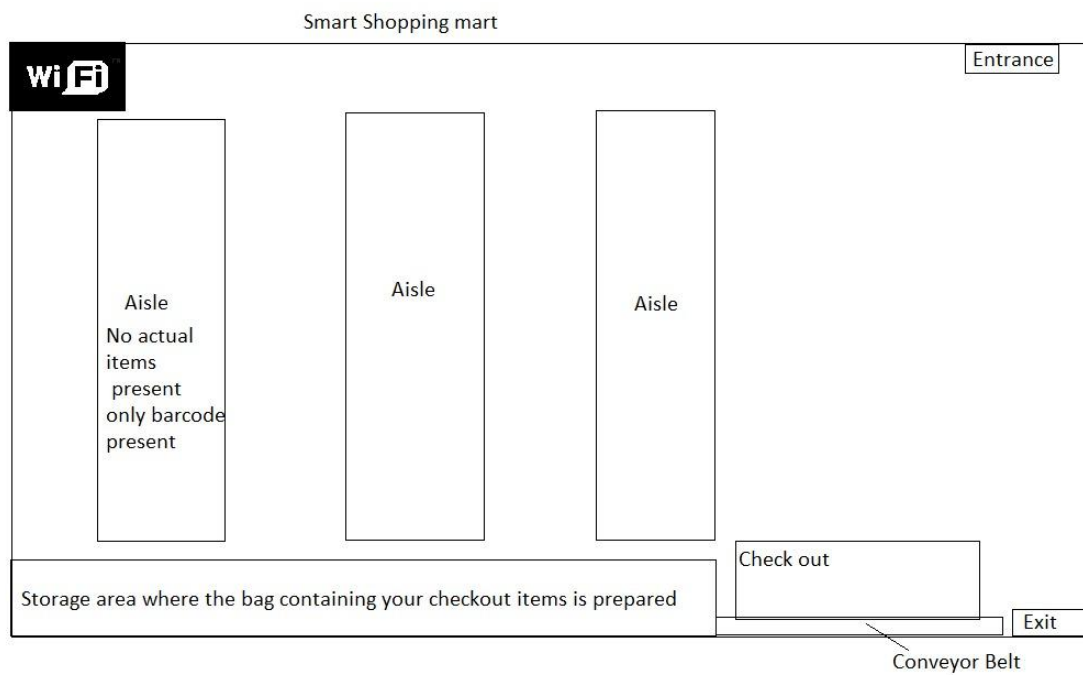
Group Members: Omer Ali Rastgar, M. Ali Raza, Zaeem Baig

## **Smart Autonomous Shopping Mart**

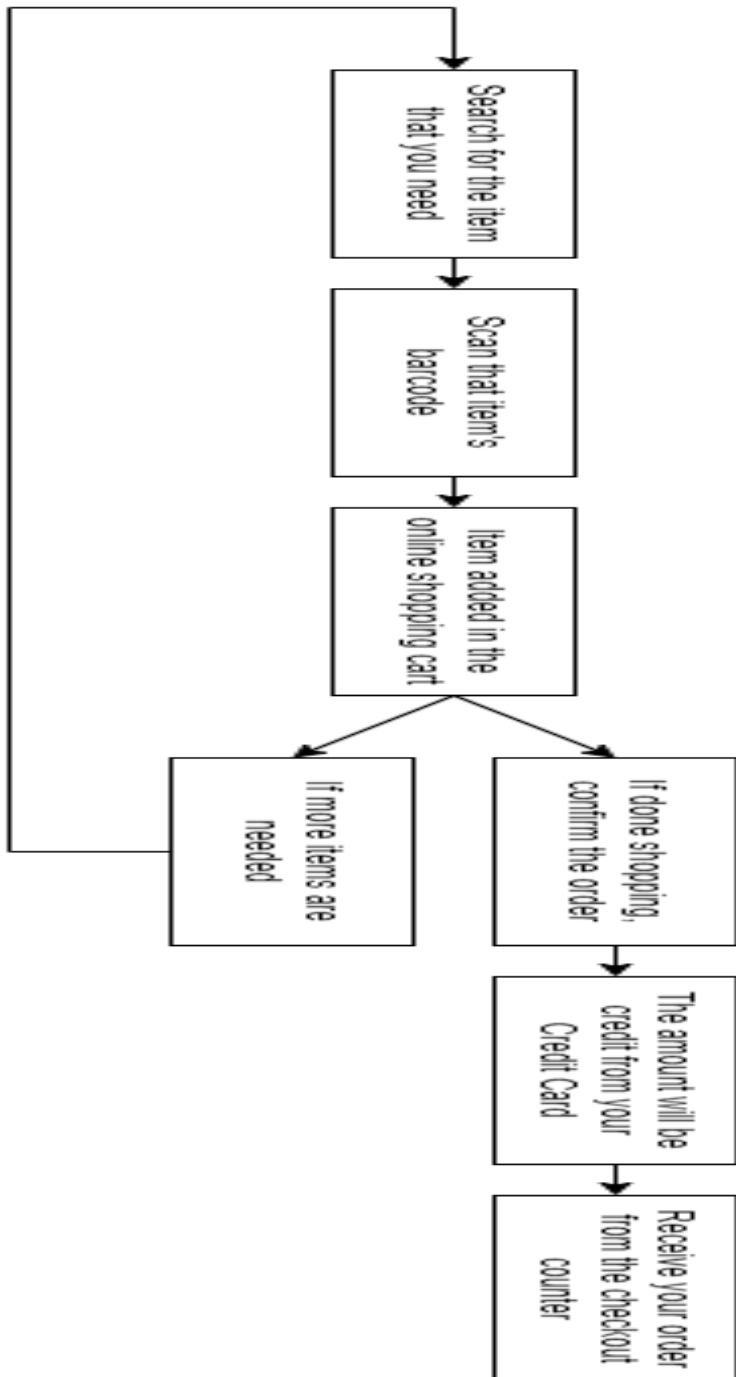
### **Theme:**

Imagine going to a shopping mart, having to move around a trolley among a bunch of people, going through each aisle and putting down all the essentials that you need. The thought of COVID enters your mind and you realize how many people must have touched the trolley handle. Not just that but even the items that you are picking out must have the bacterias all over them. After you are done shopping, you get to the checkout counter, only to see a long lane of people, practicing no social distancing.

The idea of our project is to design a new way to shop. The customers will enter the mart, scan the barcodes of the products that they would need in order to add it to the online shopping cart, confirm that they want to end their shopping and finally receive their order from the checkout.



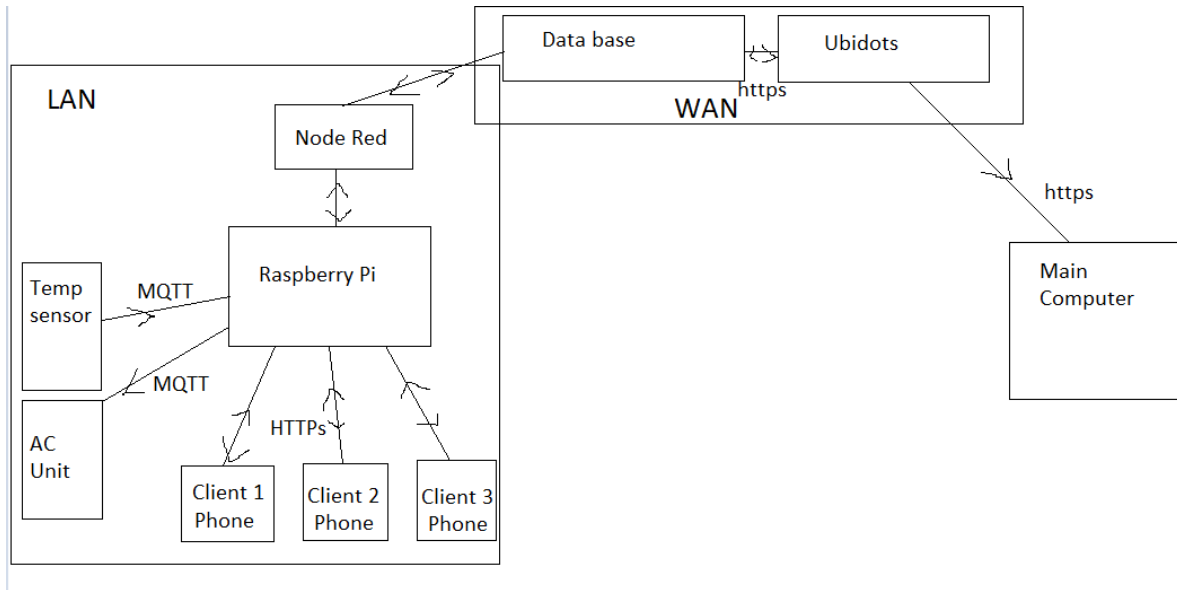
**Block Diagram:**



**Fig: process of usability for client**

## Protocols being used

- HTTPS
- MQTT



**Fig: Communication Diagram**

## Code

# Sender

**sender.py**

```
import numpy as np
import cv2
import time
import json
import base64
import requests

def send_image(img):
    #Convert image to sendable format and store in JSON
    _, encimg = cv2.imencode(".png ", img)
    img_str = encimg.tostring()
    img_byte = base64.b64encode(img_str).decode("utf-8")
    img_json = json.dumps({'image': img_byte}).encode('utf-8')

    #Send HTTP request
    response = requests.post("http://localhost:8080/save", data=img_json)
    print('{0} {1}'.format(response.status_code, json.loads(response.text)["message"]))

if __name__ == '__main__':
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FPS, 30)
    i = 0
    while True:
        _, img = cap.read()
        if i % 5 == 0:
            send_image(img)
        i += 1
```

**Fig: shows code for receiver sending image data as Json base64 [1]**

```
import os
import time
import sys
import Adafruit_DHT as dht
import paho.mqtt.client as mqtt
import json
```

```
THINGSBOARD_HOST = 'demo.thingsboard.io'
ACCESS_TOKEN = 'DHT22_DEMO_TOKEN'
```

```
# Data capture and upload interval in seconds. Less interval will eventually hang the DHT22.
```

```
INTERVAL=2
```

```

sensor_data = {'temperature': 0, 'humidity': 0}

next_reading = time.time()

client = mqtt.Client()

# Set access token
client.username_pw_set(ACCESS_TOKEN)

# Connect to ThingsBoard using default MQTT port and 60 seconds keepalive interval
client.connect(THINGSBOARD_HOST, 1883, 60)

client.loop_start()

try:
    while True:
        humidity, temperature = dht.read_retry(dht.DHT22, 4)
        humidity = round(humidity, 2)
        temperature = round(temperature, 2)
        print(u"Temperature: {:g}\u00b0C, Humidity: {:g}%".format(temperature,
humidity))
        sensor_data['temperature'] = temperature
        sensor_data['humidity'] = humidity

        # Sending humidity and temperature data to ThingsBoard
        client.publish('v1/devices/me/telemetry', json.dumps(sensor_data), 1)

        next_reading += INTERVAL
        sleep_time = next_reading-time.time()
        if sleep_time > 0:
            time.sleep(sleep_time)
except KeyboardInterrupt:
    pass

client.loop_stop()
client.disconnect()

```

Fig: Proposed code for Raspberry Pi for temperature sensor MQTT protocol [2]

## Dataset

- Generated by the owner with respect to products introduced to the shop. The quantity is stored.

### Proposed Database

- SQLite

**Dashboard 1:** Designed for clients, here clients will enter their data and scan barcode using the image camera option. The data will be processed at the back end and sent to ubidots and the database.

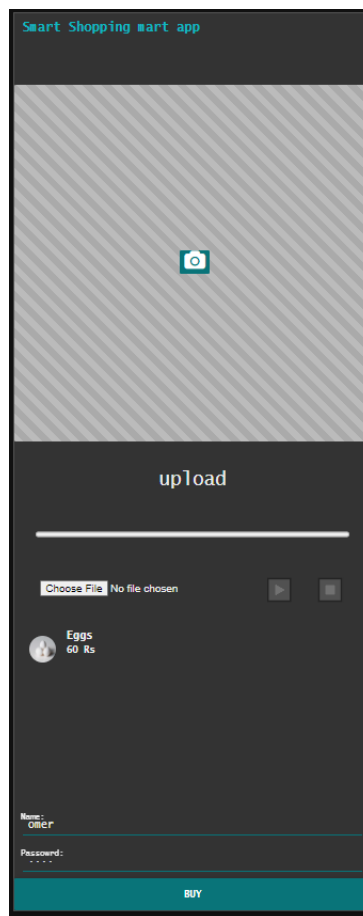
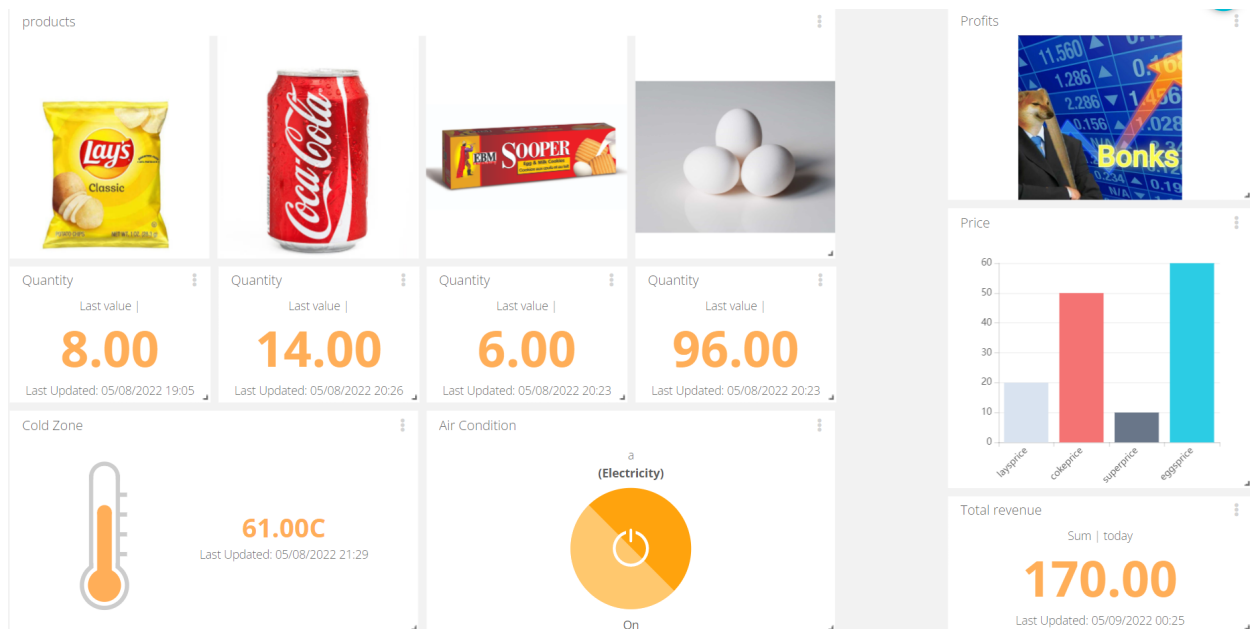


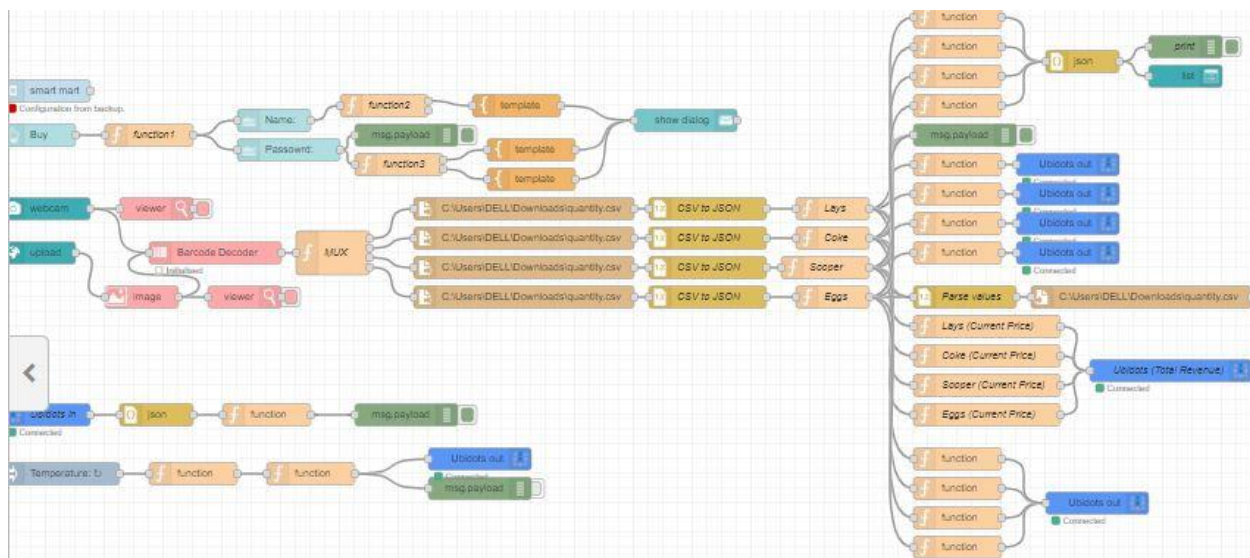
Fig: Dashboard 1

**Dashboard 2:** This is the owner's dashboard. The person can control the AC as well as see the quantity of items and total revenue made in a day.



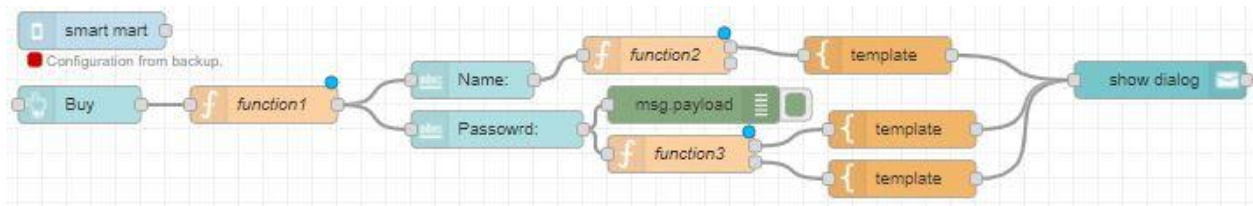
**Fig: Dashboard 2**

### Overall Flow:





## **Module 1:**



### **Nodes:**

Smart Mart: Provides remote access to an external device.

Buy: Button to trigger username and password

Function1: Enables name and password by sending an empty string.

Name: A text input.

Password: A text input.

Function2: Checks if the name entered is correct or not.

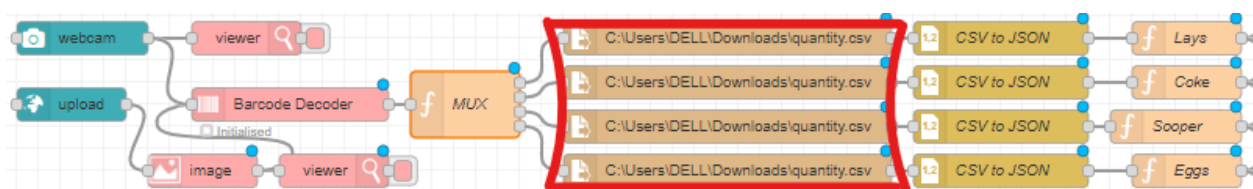
Function3: Checks if the password entered is correct or not.

Template: From top to bottom, the templates show the wrong username, wrong password or (in case of correct credentials) item bought, respectively.

Show dialog: Prints the template on the screen.

## **Module 2:**

### **Part 1:**



### **Nodes:**

Webcam: Allows us to take a picture of the barcode in PNG.

Viewer: Converts image to JSON.

Upload: Uploads the barcode picture in binary to send it to the decoder.

Image: Converts image to JSON.

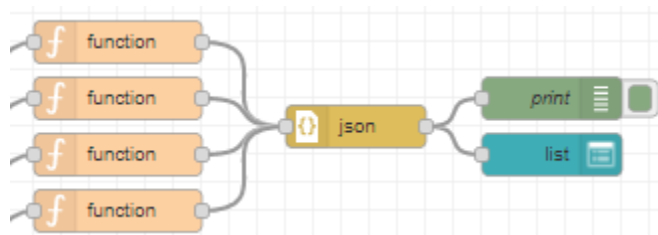
Barcode Decoder: Decodes the barcode.

MUX: function that checks which item does the barcode belong to.

Read File (Red box): Fetches and reads the data from the CSV file.

CSV to JSON: Converts the CSV file to JSON object.

Part 2:



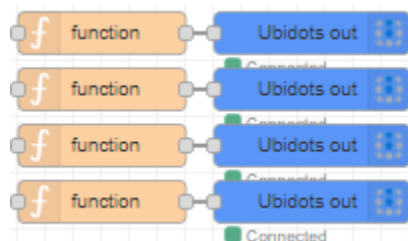
Nodes:

Function: These functions save the item bought as Lays, Coke, Sooper and eggs, depending on the one that has been bought. They contain a HTTP Get request for image transfer to node red from a given server.

JSON: Ensuring that we have a JSON object.

List: Displays the item that has been bought.

Part 3:

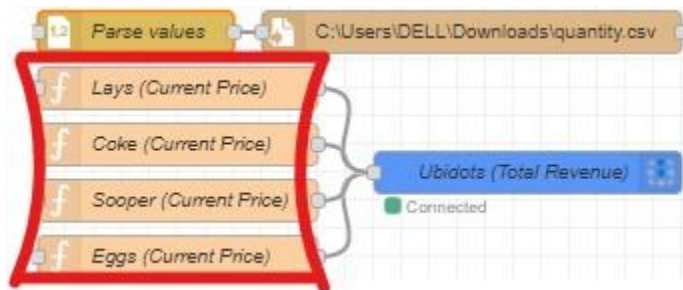


Nodes:

Function: These functions save the item bought as Lays, Coke, Sooper and eggs, depending on the one that has been bought.

Ubidots out: Send the name to Ubidots using MQTT protocol. Here Node-Red acts as a publisher while Ubidots acts as a subscriber. Data is shared through token confirmation.

Part 4:



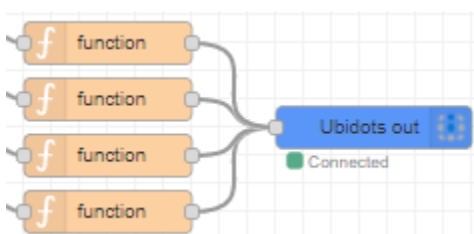
Nodes:

Parse values: Turns data into JSON objects.

Current Prices (Redbox): these functions save the current price of the items.

Ubidots (Total Revenue): This sends the price to Ubidots using MQTT protocol which is then added to the revenue generated. Here Node-Red acts as a publisher while Ubidots acts as a subscriber. Data is shared through token confirmation.

Part 5:

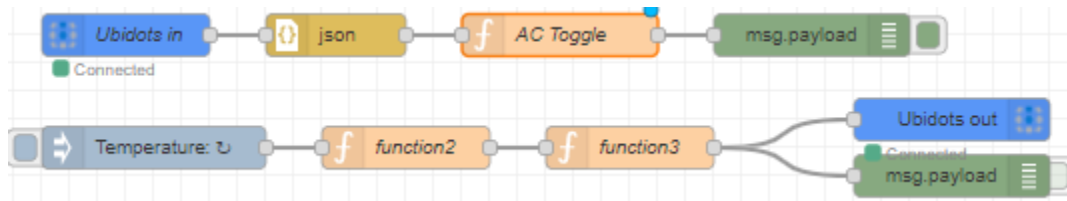


Nodes:

Functions: They save the price of the item.

Ubidots out: Sends to Ubidots using MQTT protocol. Here Node-Red acts as a publisher while Ubidots acts as a subscriber. Data is shared through token confirmation.

### **Module 3:**



### **Nodes:**

Ubidots in: The data is received from Ubidots using MQTT protocol. Here Node-Red acts as a subscriber while Ubidots acts as a publisher. Data is shared through token confirmation.

JSON: Turns data into JSON objects.

AC Toggle: Turns the AC on if the value is 1.


Temperature: Triggers the temperature.

Function2: Generates random temperature.

Function3: Saves a temperature.

Ubidots out: Sends to Ubidots using MQTT protocol. Here Node-Red acts as a publisher while Ubidots acts as a subscriber. Data is shared through token confirmation. Node red library for ubi dots is the brokerer


### **Email generation using Ubidots event handler**

lays alert!  Inbox x



**Notifications Ubidots** <service@ubidots.com>

to me ▾

 English ▾ > Urdu ▾ [Translate message](#)

Hey there, lays was 0.0 at 2022-05-11 02:44:10 +0500.

 Reply

 Forward

### **Concepts learnt:**

TLS protocol

- Https: private and public keys: get and post method.
- MQTT: subscriber publish model (Token generation )

### **Task distribution:**

Ali Raza: NodeRed design flow

Omer Rastgar: Troubleshooting and general workflow

Zaeem baig: Integration of Ubidots with node red

## Bar code:



**Fig: Images used for camera scanning, made using barcode generator [3]**

## Libraries being used:

1. Node-red-contrib-image-tools
2. Node-red-contrib-remote
3. Node-red-contrib-ui-upload
4. Node-red-dashboard
5. Node-red-node-ui-webcam
6. ubidots-nodered

## Link:

[https://stem.ubidots.com/app/dashboards/public/dashboard/2k8Jn50Akdml2kykNC2Kc8En6yeVL\\_OtUvWFvJhpbm8?from=1651921296854&to=1652554799999&datePicker=true&nonavbar=true](https://stem.ubidots.com/app/dashboards/public/dashboard/2k8Jn50Akdml2kykNC2Kc8En6yeVL_OtUvWFvJhpbm8?from=1651921296854&to=1652554799999&datePicker=true&nonavbar=true)

## References

1. <https://linuxtut.com/en/8da21f52e379469d744b/>
2. <https://thingsboard.io/docs/samples/raspberry/temperature/>
3. <https://barcode.tec-it.com/en/Code128>

