IoT Coursework

Student: up2097531

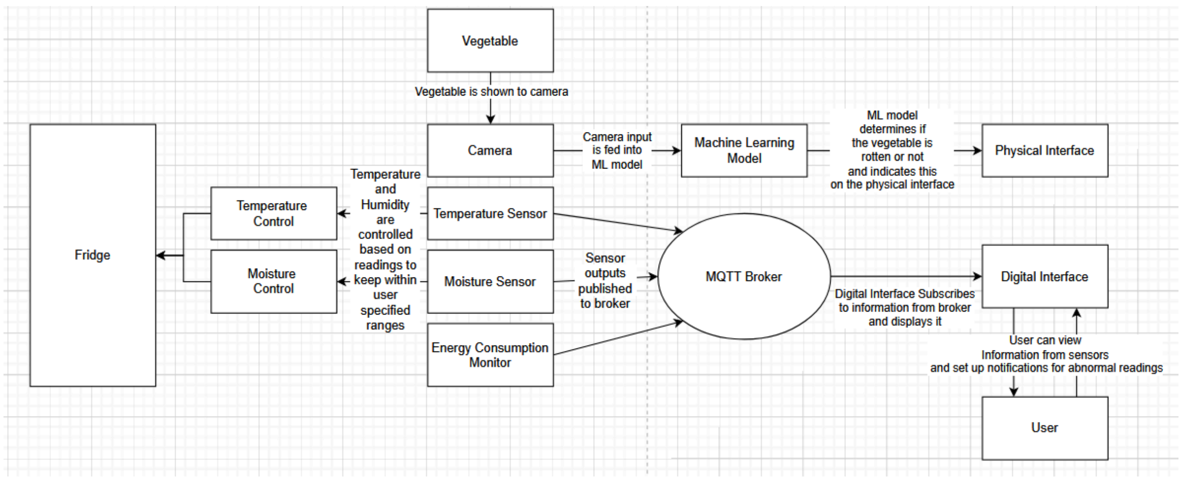
1. Introduction

The aim of this project is to create a smart refrigerator that will reduce food waste. Food waste is a huge problem in the UK with around 9.5 million tonnes of food being thrown away each year, almost half of that still being edible (Graham, 2024). The planned device will help people detect if their food is still edible to reduce the amount of edible food waste and automatically monitor and change the environment in which their food is stored so it will last longer.

The development was structured as follows:

- Design
  - Create design for whole system
  - Find components that achieve system design
- Implementation
  - Fruit Classification
    - Create and train Classification model
    - Test model
    - Apply live feed input
    - Create interface for output
  - Environment monitoring
    - Connect sensors and control interfaces
    - Write software for components
    - Create MQTT broker interface
- Conclusion
  - Evaluate Implementation against aim
  - Evaluate cost of implementation

2. Design

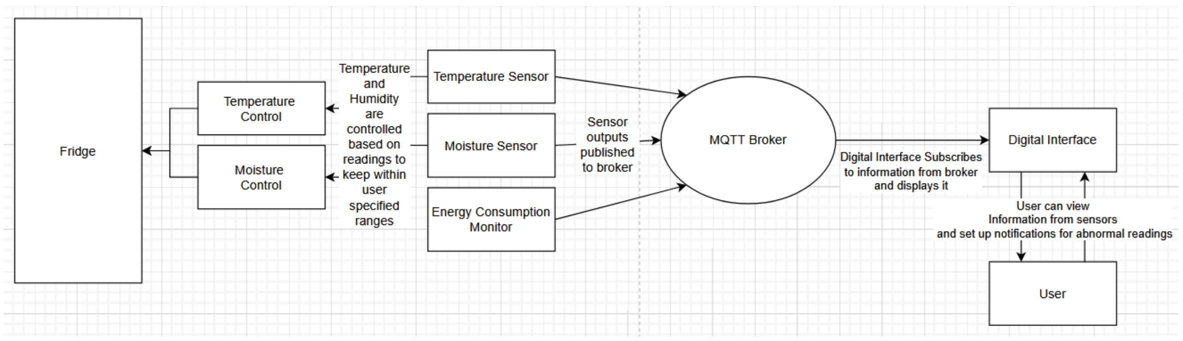


Vegetable
Vegetable is shown to camera
Camera
Camera input is fed into ML model
Machine Learning Model
ML model determines if the vegetable is rotten or not and indicates this on the physical interface
Physical Interface
Fridge
Temperature Control
Moisture Control
Temperature and Humidity are controlled based on readings to keep within user specified ranges
Temperature Sensor
Moisture Sensor
Energy Consumption Monitor
Sensor outputs published to broker
MQTT Broker
Digital Interface
Digital Interface Subscribes to information from broker and displays it
User can view Information from sensors and set up notifications for abnormal readings
User

Fig 1 - Whole system design

There will be two smaller devices as part of the IoT solution. The first will be placed inside the refrigerator where it will monitor and control the temperature, moisture and energy consumption of the refrigerator. The second will be mounted on the outside of the refrigerator which will take pictures of fruit and vegetables and analyse these images using a machine learning model. Once analysis is complete the physical interface will indicate whether the food is safe to eat or not.

2.1 Inside The Refrigerator



Fridge
Temperature Control
Moisture Control
Temperature and Humidity are controlled based on readings to keep within user specified ranges
Temperature Sensor
Moisture Sensor
Energy Consumption Monitor
Sensor outputs published to broker
MQTT Broker
Digital Interface
Digital Interface Subscribes to information from broker and displays it
User can view Information from sensors and set up notifications for abnormal readings
User

Fig 2 - Design for the environment control device

The device inside of the refrigerator consists of a DHT11 temperature and humidity sensor (Mouser Electronics, 2012), a 3-6v motor with fan blade to represent the temperature control and an LED to represent the humidity control. Separately a PZEM-004T voltage multimeter is included to monitor the energy consumption of the refrigerator itself (Innovators Guru, 2019).

The DHT11 sensor has been chosen as they are low cost with a temperature range of 0°c to 50°c and a humidity range of 20% to 90% which both cover the ranges of temperature and humidity this device will encounter. The multimeter can monitor voltages up to 280V and amperage up to 100A which again is within the range of the average refrigerator (Holley, 2025).

As there is no access to the temperature and humidity control within the refrigerator the 3-6v motor will represent the temperature control and an LED will represent the humidity control. The speed of the motor and brightness of the LED will change based on the sensor readings to indicate action being taken based on the climate of the refrigerator.

All sensor readings will be published to an AdaFruit Broker (Rubell, 2019) where the user can monitor the readings from the sensors graphically. Using AdaFruit means that a dedicated custom interface will not need to be created; however if a more customised interface was required this could be done. From the interface the user will be able to specify temperature and humidity ranges for the sensors to keep the refrigerator environment within.

2.2 Outside The Refrigerator

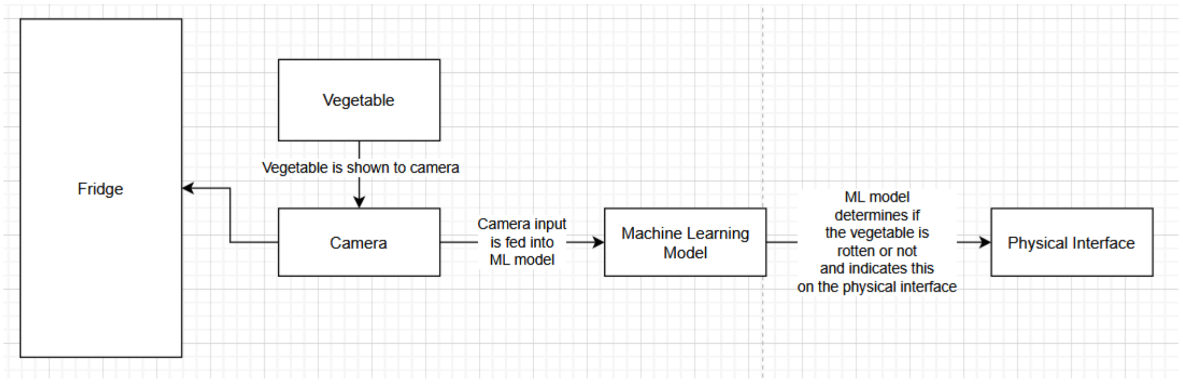

Fridge
Vegetable
Vegetable is shown to camera
Camera
Camera input is fed into ML model
Machine Learning Model
ML model determines if the vegetable is rotten or not and indicates this on the physical interface
Physical Interface

Fig 3 - Design for the fruit state detection device

The device mounted outside of the refrigerator will consist of a camera, a computer to run the classification model on and a physical interface which will indicate if the food shown to the camera is safe to eat or not.

The camera and computer will be a laptop which will be running a KNN classification model built in python. This can be reduced to a camera module and python script running on a raspberry pi to make it mountable and would be cheaper to create.

For the classification model, TensorFlow with Keras (TensorFlow, 2019) will be used to create a custom trained algorithm. For training, the fresh, rotten and formalin-mixed fruits dataset created by Bijoy et al. (2024) which contains over 60,000 images of fruits with varying quality will be used. This dataset provides a large set of images, raw and edited, which will provide a large variation of data for the model. The dataset also includes six different types of fruit which will help with the versatility of the model application.

The result of the classification will be indicated using two LEDs green and red. The green LED will activate if the fruit is safe to consume and the red LED will activate when the fruit is identified as rotten. This interface is very low cost and low power which balances out the relatively high cost and power consumption of the rest of the device. It also is a clear indicator for everyone as it does not require language comprehension and doesn't necessarily rely on colour.

3. Implementation

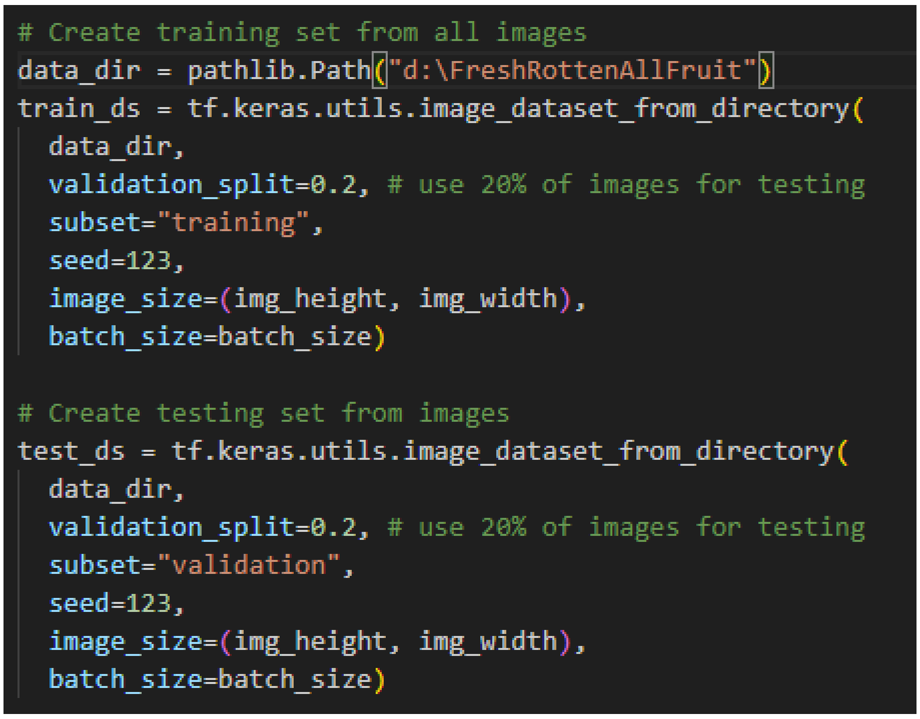
3.1 Fruit Classification Device

3.1.1 Creating the model

As the module was two parts, so was the implementation. As the fruit detection system would take the longest this was the priority. Before any physical implementation an algorithm for detecting the state of fruit needed to be developed.

The detection algorithm uses the Keras Sequential Model (TensorFlow, 2025). As the algorithm needed was highly specific, it was created from scratch. To do this a python program was created.

This first loads the dataset and splits it into training and testing sets using an 80:20 split respectively (Fig 4). The datasets themselves are split into batches of 10 images. This means that each time an image is needed by the model 10 images are loaded into memory as opposed to the full dataset decreasing computation load.



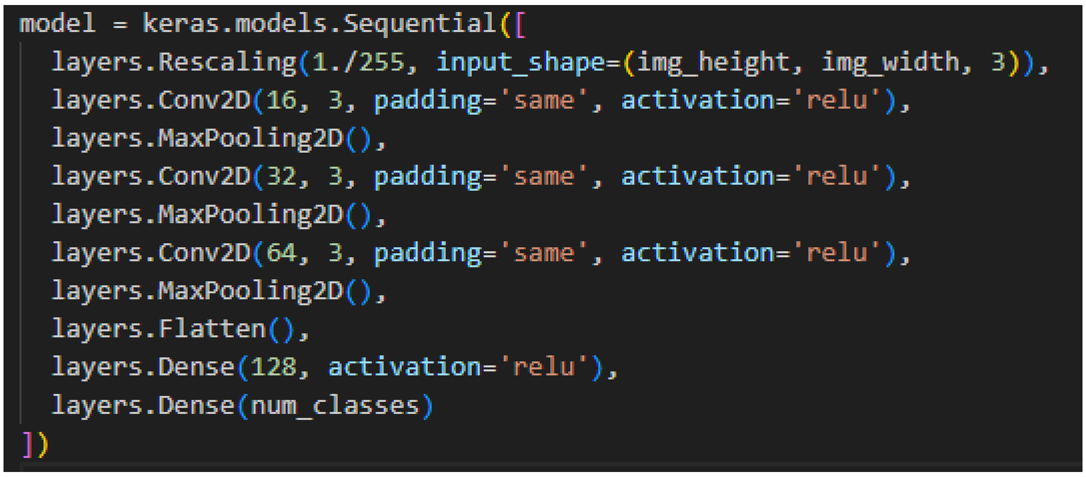
```
# Create training set from all images
data_dir = pathlib.Path("d:\FreshRottenAllFruit")
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2, # use 20% of images for testing
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

# Create testing set from images
test_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2, # use 20% of images for testing
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

Fig 4 - Code for creating the training and testing datasets

From here the program then defines the model. Keras Sequential is a simple neural network that uses layers to identify images. This was used as the time to train and computational load would be low and the model produced at the end of training would also be very low profile. This came at the cost of accuracy. The model architecture used to train this algorithm is as follows (Fig 5):

- Layer 1 - 16 filters are applied to the image. This helps identify the edges of the object.
- Layer 2 - 32 filtered are applied to the image. This helps define more complex features such as texture or shape, improving classification.
- Layer 3 - 64 filters are applied to the image. Increases definition complexity.
- Layer 4 - 128 filters are applied to the image. The final filter layer which tries to identify full objects by combining features.

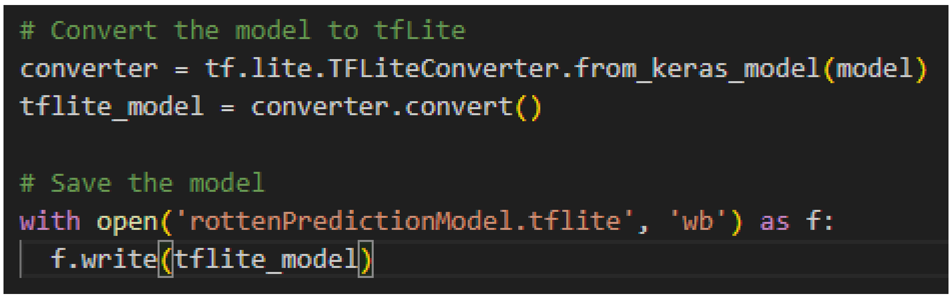
Once all layers have been applied to the image the model guesses what class the image belongs to.



```python
model = keras.models.Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Fig 5 - Model architecture

Once training has completed the model is converted to the ‘tflite’ format for use in the embedded system.



```python
# Convert the model to tfLite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model
with open('rottenPredictionModel.tflite', 'wb') as f:
  f.write(tflite_model)
```

Fig 6 - Convert model to tflite

3.1.2 Limitations

As it stands the model is not good at classifying real word data. Upon training the model scored an accuracy rating of 0.98 out of 1. At face value this is a very good score however when testing against new data the model tends to struggle. This is most likely due to lack of variance in the training and testing dataset. The vast majority of images within the dataset were taken on a white background with perfect lighting (Fig 7).

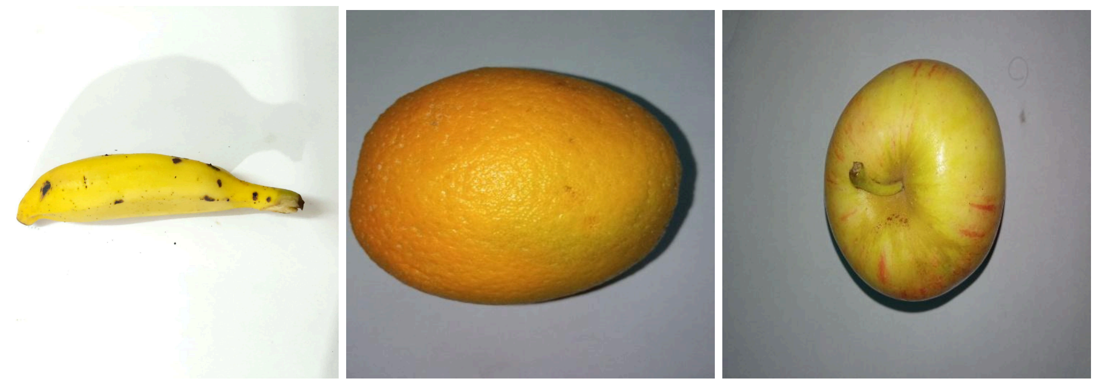


Fig 7 - Examples of images within the dataset

Because of this, images used that aren't in these conditions are harder for the model to classify. To improve the accuracy a more varied set of environments should be used as well as the existing one.

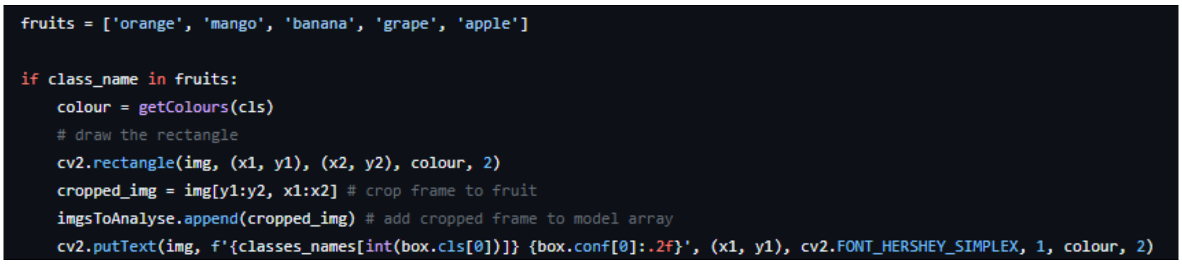
Another factor that may increase accuracy would be to include more layers within the model architecture. This would allow the algorithm to learn with more detail at the cost of computational load and time.

The model has also only been trained on fruit. Although detection of rotten vegetables may be possible if they share similar characteristics to rotten fruit it is not guaranteed and will most likely be very inaccurate.

3.1.2 Using the model

Now that a functional classification algorithm has been created it can be used within the embedded system. First there needed to be a way to capture images of the fruit from the refrigerator.

Using a camera and the cv2 python library (OpenCV, 2025) a live feed can be used as an input. The raw input cannot be used directly in the model as analysing every frame of the feed is inefficient therefore another identification model is needed to first find fruit in the camera frame. A quick solution to this was YOLOv8 (Ultralytics, 2023) a pre-built classification model that can quickly identify objects in real time which saves the need to create one for this specific purpose. The downside to this however is that the algorithm here will also detect other objects within the frame. As YOLOv8 is relatively high level we can define what objects to crop when found however there will still be increased computation due to other detections (Fig 8).



```python
fruits = ['orange', 'mango', 'banana', 'grape', 'apple']

if class_name in fruits:
    colour = getColours(cls)
    # draw the rectangle
    cv2.rectangle(img, (x1, y1), (x2, y2), colour, 2)
    cropped_img = img[y1:y2, x1:x2] # crop frame to fruit
    imgsToAnalyse.append(cropped_img) # add cropped frame to model array
    cv2.putText(img, f'{classes_names[int(box.cls[0])]} {box.conf[0]:.2f}', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 1, colour, 2)
```

Fig 8 - High level detection of defined fruits inside object detection function

The function for image collection works by first opening a live camera feed. The feed is then fed into the YOLO8 algorithm which detects all objects within the frame (Fig 9). If the objects in the frame are identified as any of the defined fruit, the function will crop that frame and add it to an input array (Fig 10).

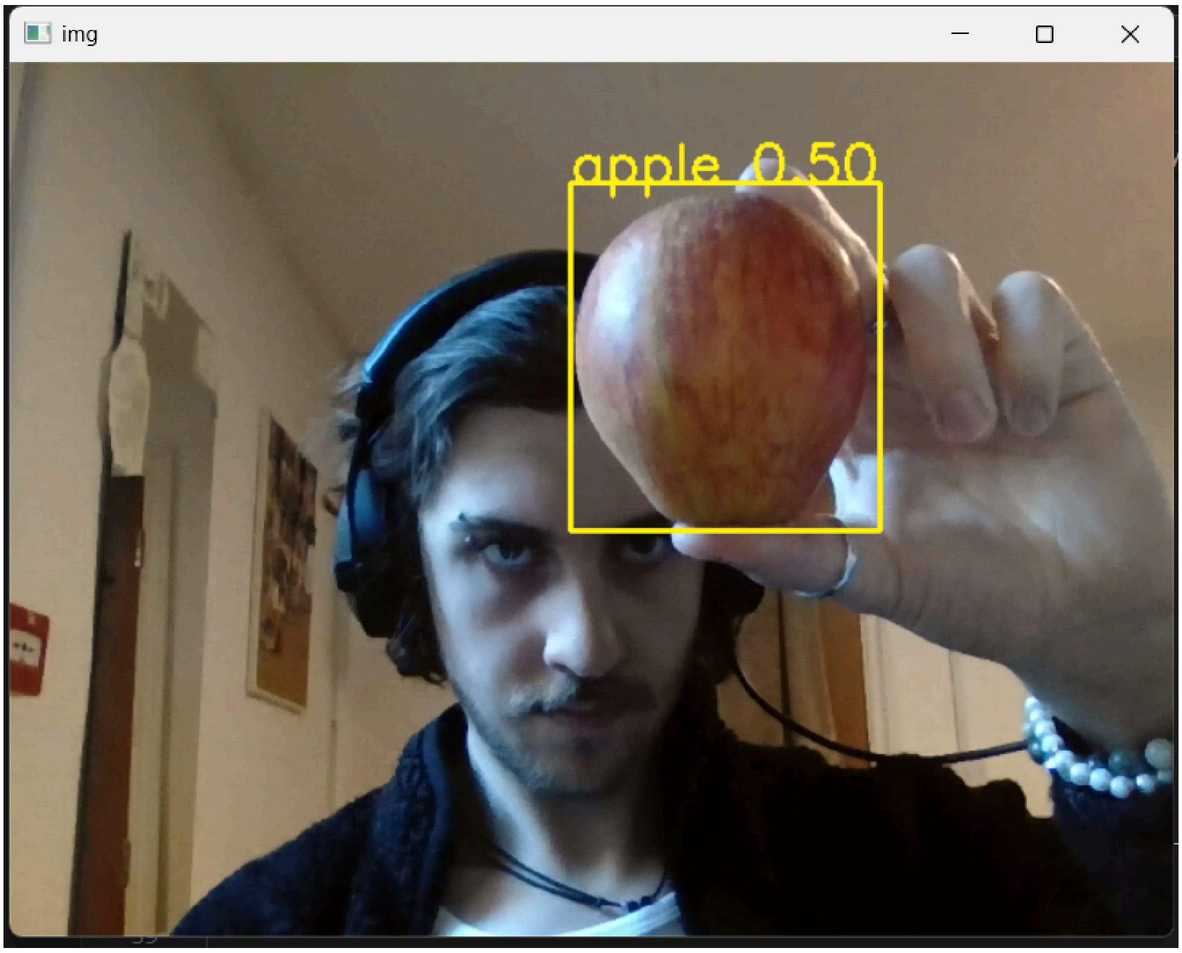
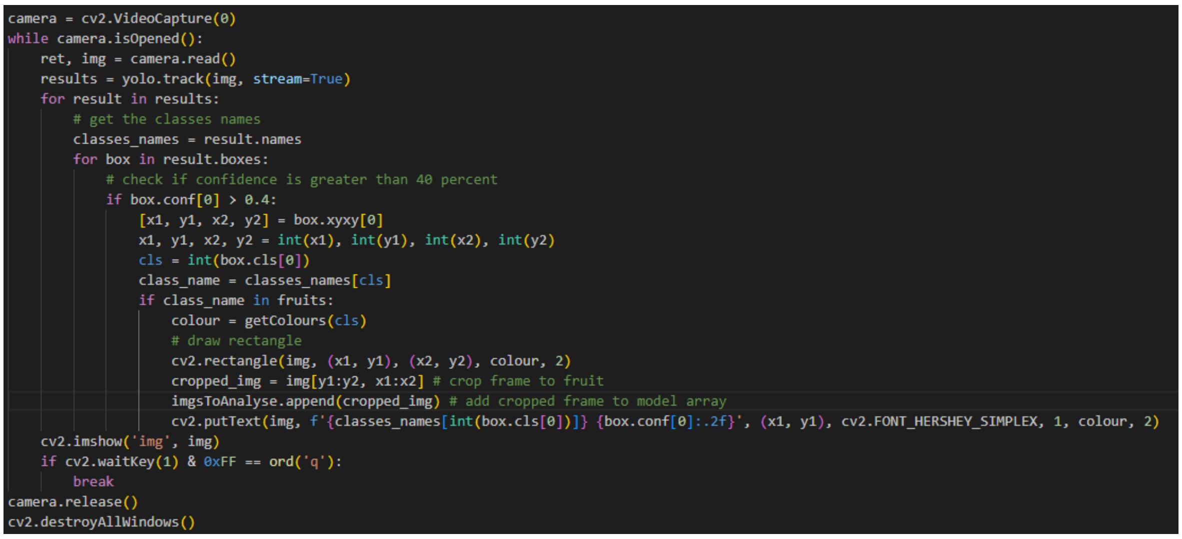


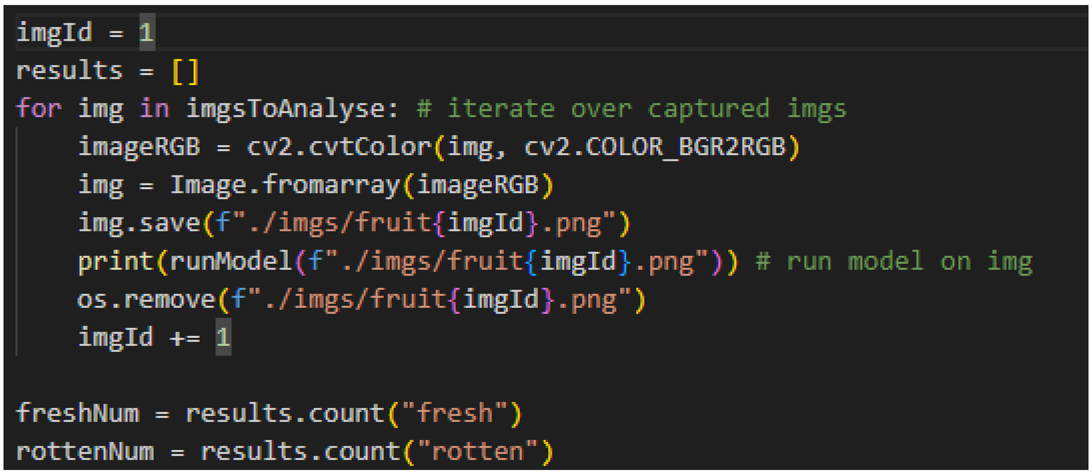
Fig 9 - Camera feed Running YOLOv8 to detect fruit



```python
camera = cv2.VideoCapture(0)
while camera.isOpened():
    ret, img = camera.read()
    results = yolo.track(img, stream=True)
    for result in results:
        # get the classes names
        classes_names = result.names
        for box in result.boxes:
            # check if confidence is greater than 40 percent
            if box.conf[0] > 0.4:
                [x1, y1, x2, y2] = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                cls = int(box.cls[0])
                class_name = classes_names[cls]
                if class_name in fruits:
                    colour = getColours(cls)
                    # draw rectangle
                    cv2.rectangle(img, (x1, y1), (x2, y2), colour, 2)
                    cropped_img = img[y1:y2, x1:x2] # crop frame to fruit
                    imgsToAnalyse.append(cropped_img) # add cropped frame to model array
                    cv2.putText(img, f'{classes_names[int(box.cls[0])]} {box.conf[0]:.2f}', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 1, colour, 2)
    cv2.imshow('img', img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
camera.release()
cv2.destroyAllWindows()
```

Fig 10 - Object detection function and fruit cropping

Once the camera capture is completed the input array is iterated over and each image is analysed by the tflite model. A classification is made on every image of the array and put into a list where a count of each class is made. The class with the highest count is the overruling class making it the final classification of that fruit (Fig 11).

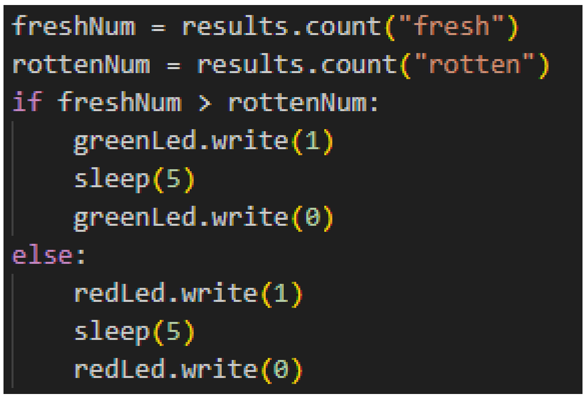


```python
imgId = 1
results = []
for img in imgsToAnalyse: # iterate over captured imgs
    imageRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(imageRGB)
    img.save(f"./imgs/fruit{imgId}.png")
    print(runModel(f"./imgs/fruit{imgId}.png")) # run model on img
    os.remove(f"./imgs/fruit{imgId}.png")
    imgId += 1

freshNum = results.count("fresh")
rottenNum = results.count("rotten")
```

Fig 11 - Captured images are fed into runModel() to be classified

3.1.3 Applying to hardware

Applying the model to hardware was the simplest task of this section. Using the firmata (Arduino, 2024) library the python program can be used to control an Arduino. From this once a final classification has been made the program can indicate to the user the output using a green or red LED. This is done using an if statement on the fresh and rotten classification counts (Fig 12).



```
freshNum = results.count("fresh")
rottenNum = results.count("rotten")
if freshNum > rottenNum:
    greenLed.write(1)
    sleep(5)
    greenLed.write(0)
else:
    redLed.write(1)
    sleep(5)
    redLed.write(0)
```

Fig 12 - Indicate on Interface if fruit is fresh or rotten

The circuit diagram for the hardware is as follows.

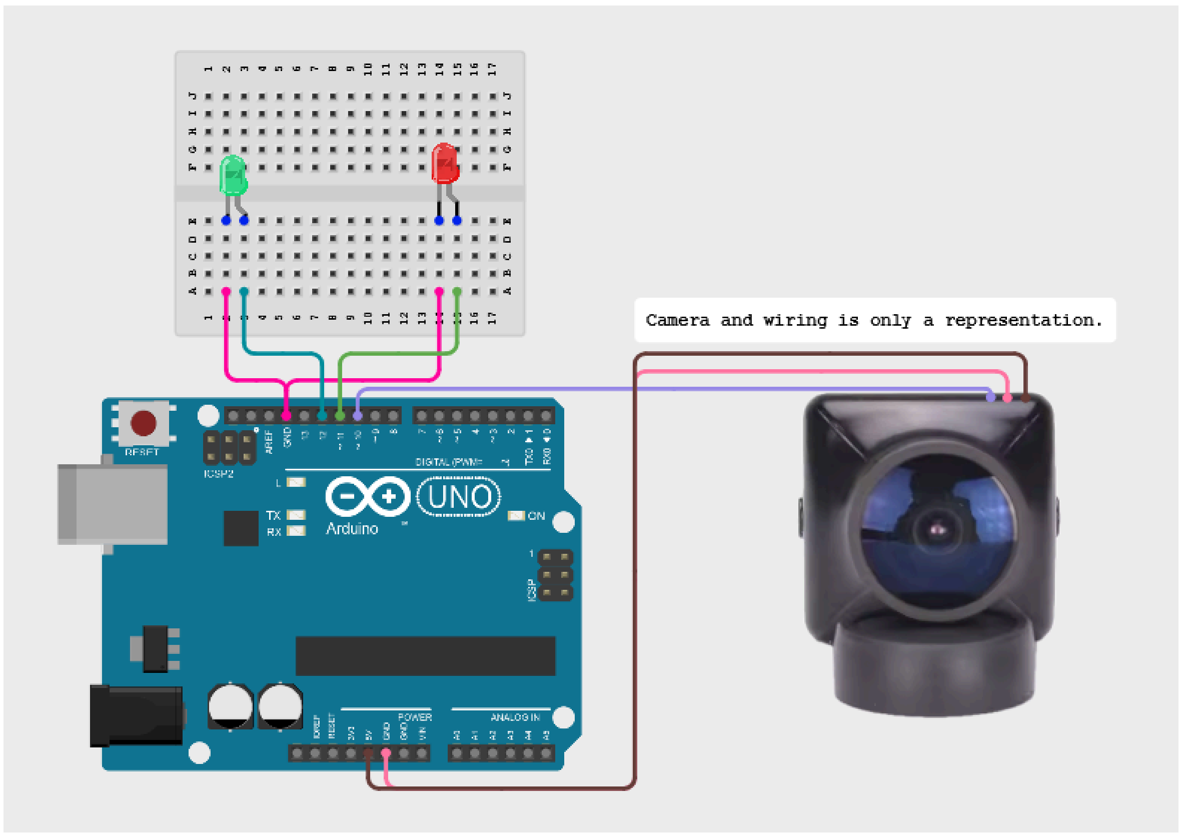


Fig 13 - Circuit diagram for the device

A simple, two LED, interface was chosen for accessibility reasons. This way no text needed to be interpreted so anyone can understand the indications.This device can help people who are visually impaired such as colour blindness or partial blindness as the LEDs shine at full brightness. Although green and red, the colour of the LED is not necessary as the position of the shining LED can indicate the response. On top of accessibility the interface does not require much power to run meaning more can be given to the computation for the classification model.

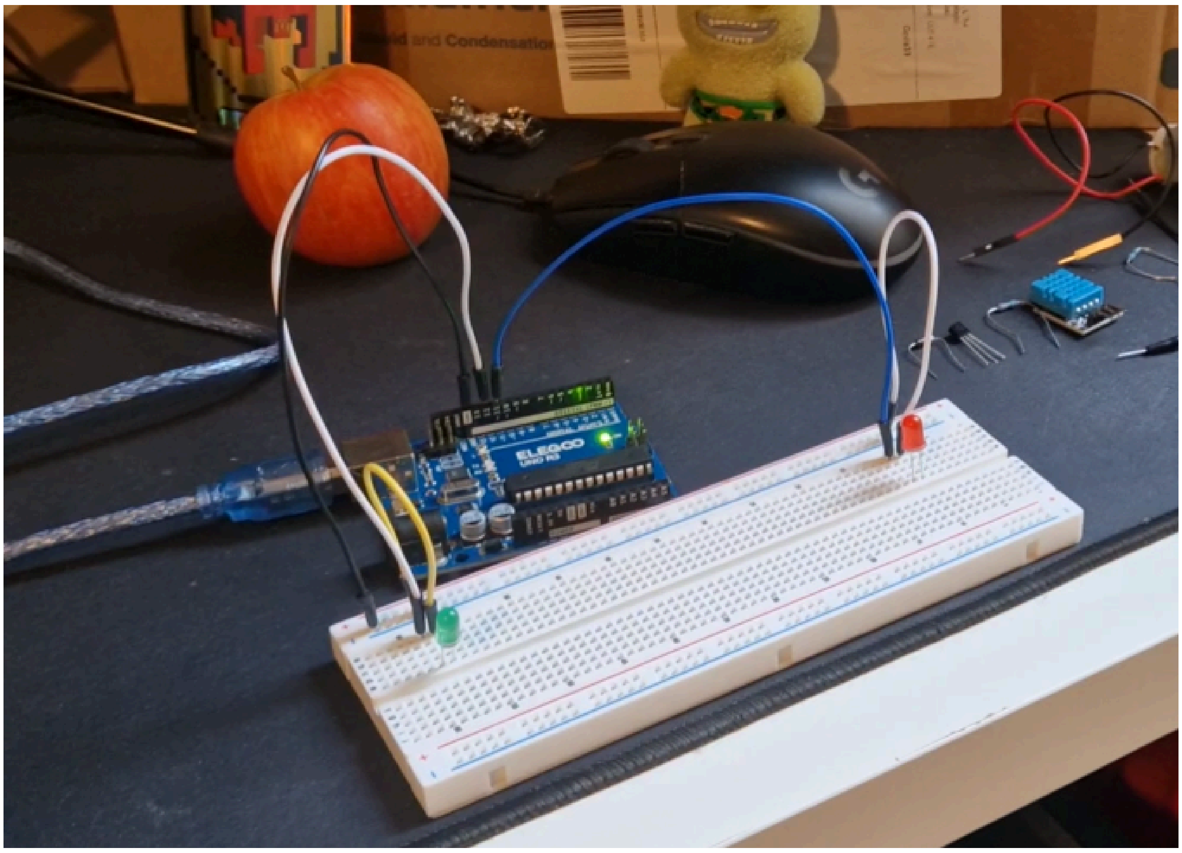


Fig 14 - Physical implementation of the interface

3.2 Environment Monitoring and Control

3.2.1 Hardware

The second part of the module relies more on hardware components to monitor and control the environment inside the refrigerator. As the temperature and humidity control for the refrigerator could not be accessed, these aspects were represented using a fan for temperature and an LED for humidity control. The circuit diagram is as follows.

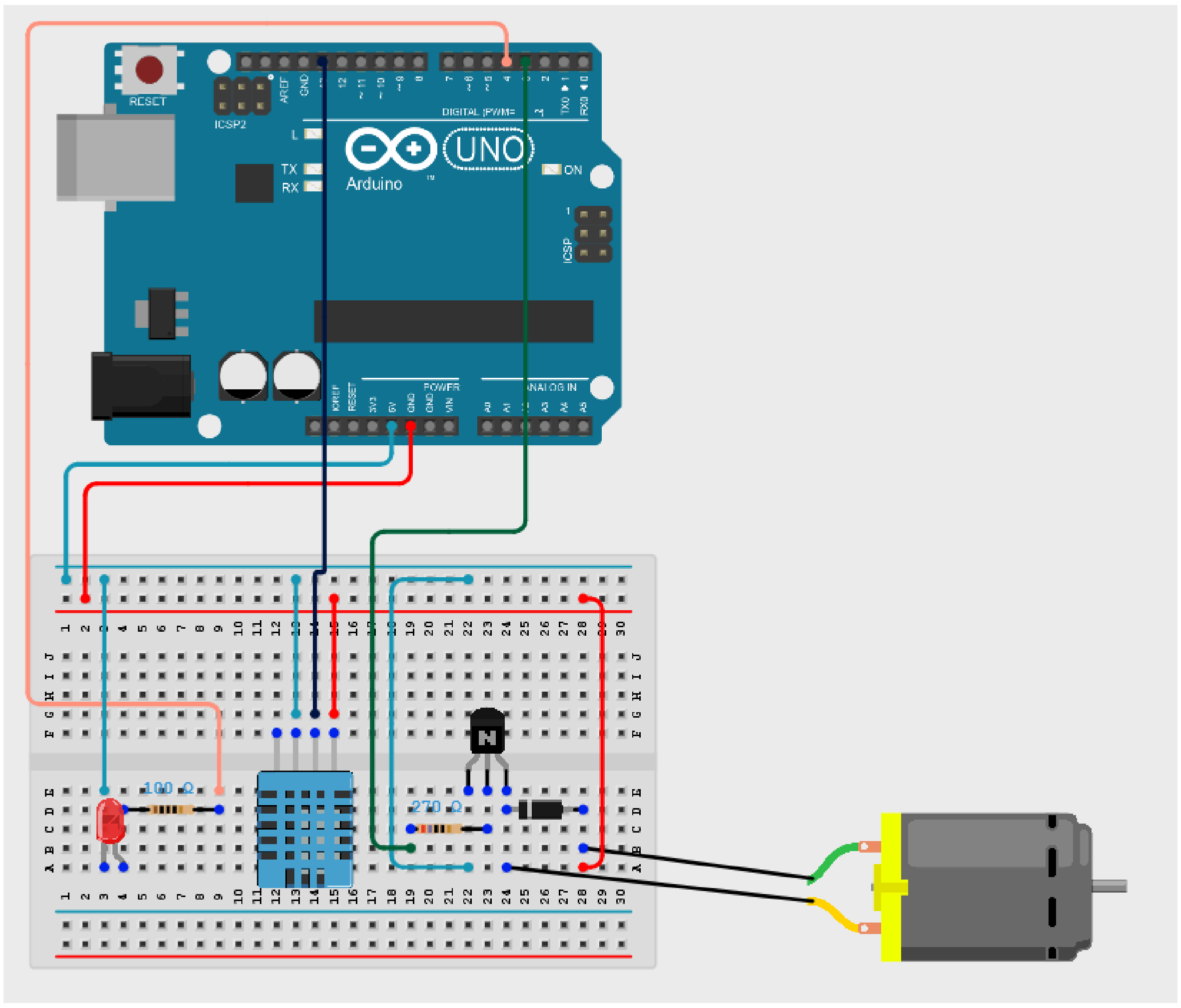


Fig 15 - Circuit diagram for the environment monitoring device

The device that is placed inside the refrigerator logs the temperature and humidity of the environment using a DHT11 temperature sensor. The arduino then reads the output of the sensor and changes the power output to the controls in accordance with the readings. The LED representing the humidity controller for the refrigerator turns on if the humidity is below 85%. As the humidity decreases past this the power output to the LED increases signalling the controller is working harder to get the humidity back to safe levels. The temperature control works in a similar way. If the temperature increases above 8 degrees celsius the fan motor will turn on. As the temperature increases, the fan speed will also increase to try and push the temperature below 8 degrees. Overall the circuit is a simple combination of three components.

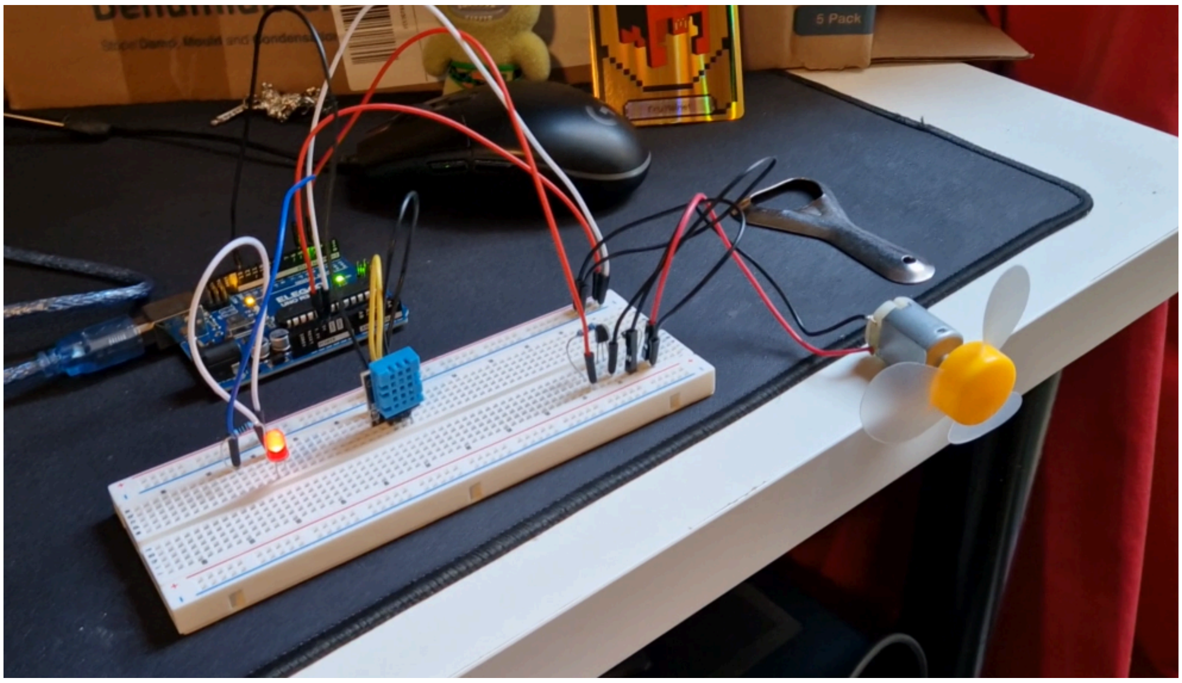
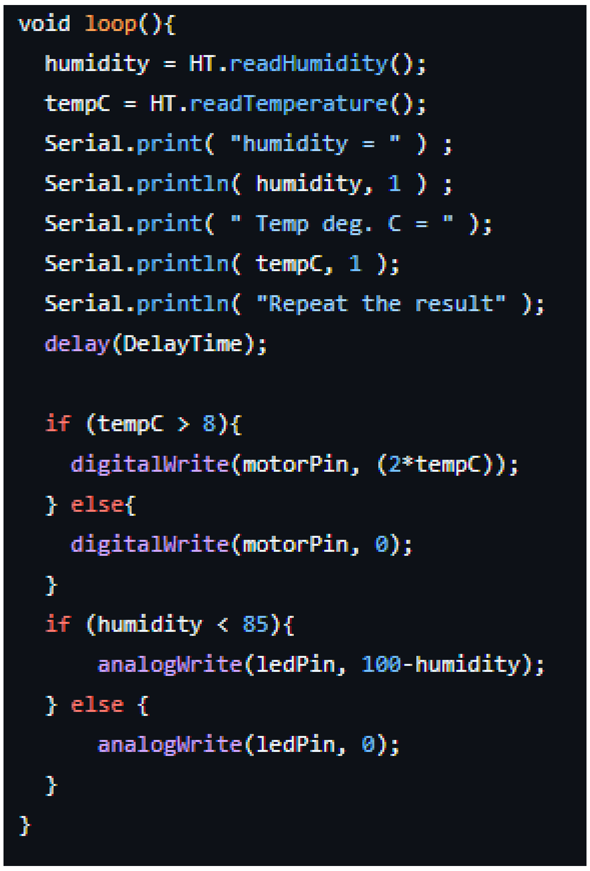


Fig 16 - Physical implementation of the environment monitoring device

3.2.2 Software

The software for the second part of the module is written directly on the arduino which is made up of two halves. The first is the initialisation and operation code for the DHT11 sensor. The second is the reactive components to the sensor readings.

The loop function is where the bulk of the software resides (Fig 17). Every four seconds the sensor takes a reading for temperature and humidity. These readings are then compared to the safe bounds for inside the refrigerator (85% humidity (Adams, 2025) and 8 degrees temperature (Gallagher, 2025)).



```
void loop(){
  humidity = HT.readHumidity();
  tempC = HT.readTemperature();
  Serial.print( "humidity = " ) ;
  Serial.println( humidity, 1 ) ;
  Serial.print( " Temp deg. C = " );
  Serial.println( tempC, 1 );
  Serial.println( "Repeat the result" );
  delay(DelayTime);

  if (tempC > 8){
    digitalWrite(motorPin, (2*tempC));
  } else{
    digitalWrite(motorPin, 0);
  }
  if (humidity < 85){
      analogWrite(ledPin, 100-humidity);
  } else {
      analogWrite(ledPin, 0);
  }
}
```

Fig 17 - Loop function for taking sensor readings and making reactions

To change the power output based on the readings of the sensor a constant is operated on based on the reading. For example the power output for the temperature reading is double the value of the temperature value. For the humidity control the output is 100 minus the humidity value meaning at 84% the output would be 16 slowly increasing as the reading decreases.

3.3 Failed Implementation

Two aspects of the design are not implemented fully. The first is the MQTT broker as the software for the ESP-01 (Ai-Thinker, 2017) component did not work. The ESP-01 component allows the arduino to communicate using WiFi which is needed to subscribe to the MQTT feed. The feed dashboards were created to receive the data using AdaFruit (Fig 18) but unfortunately the transmission couldnt be achieved.

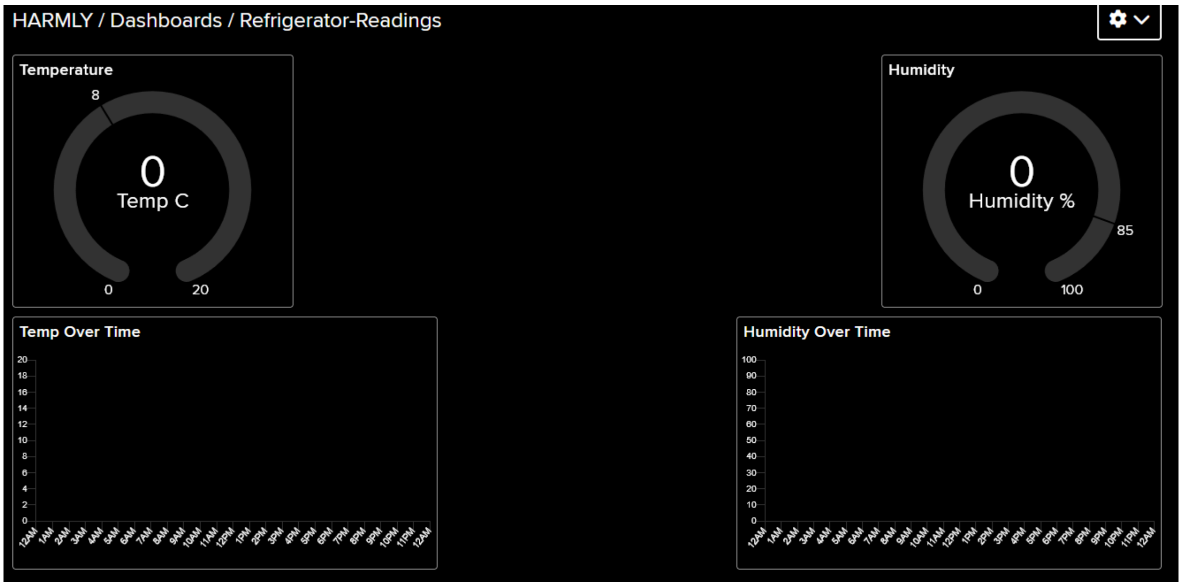


Fig 18 - Unfinished implementation of the MQTT broker dashboard

The second aspect not implemented was the energy monitoring device due to the lack of necessary components. The design planned to use the PZEM-004T multimeter however as there was no funding for the project the component could not be acquired. If implemented within the system the circuit diagram for the environment monitor would be as follows.

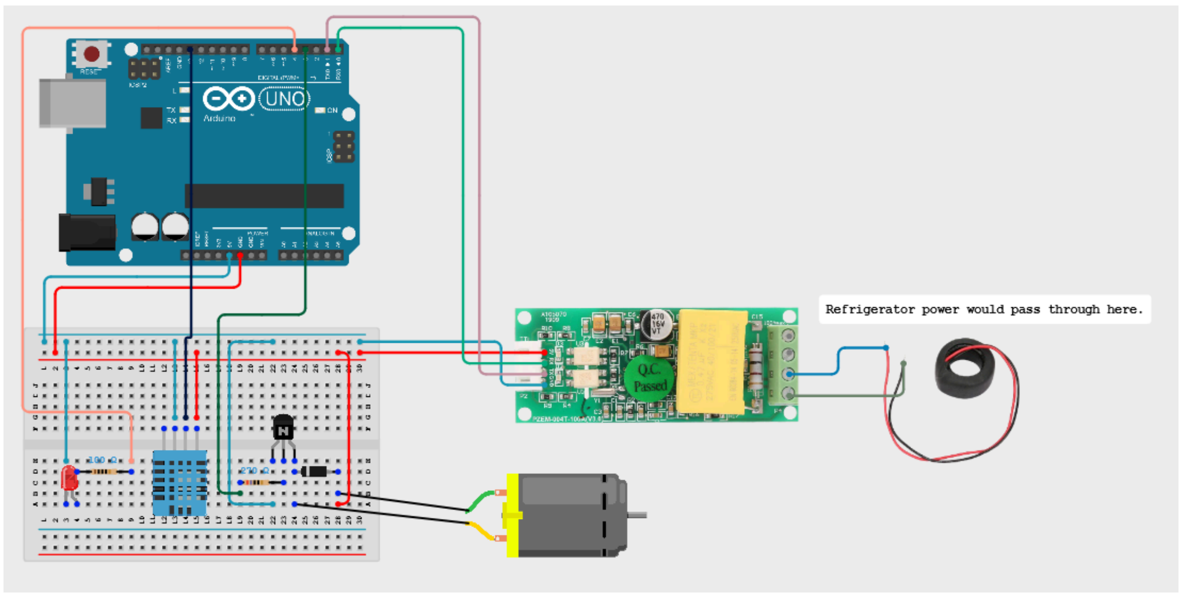


Fig 19 - Circuit diagram for environment monitoring device with PZEM-004T module

Readings could be taken directly from the arduino and published to the MQTT broker much like the temperature and humidity.

4. Conclusion

As the majority of the system has been implemented evaluation of the device can be conducted. The overall aim of the project is to help reduce food waste by helping individuals keep their refrigerator environments within safe levels and definitively identify if food is safe for consumption. The first half of the aim is achieved with the use of the environment monitor which tracks temperature and humidity. Although the implementation was not directly connected to the controls of the refrigerator it clearly illustrated how it can detect changes in the environment and provide dynamic power to the corresponding components. As this is all automated it is objectively easier for the user to keep the refrigerator environment within safe conditions than doing this manually.

The second half of the aim has somewhat been achieved with the implementation of the fruit classification device. This has only partially been achieved however as the model has a low accuracy meaning it is not reliable enough for actual usage. Another drawback of the model is that it is only trained on six types of fruit. Although other rotten food may be identifiable if they share the same characteristics as any of the rotten images the model is trained on, this cannot be relied on therefore the application of the device is narrow.
Despite these setbacks the rest of the device does operate successfully with a live camera input that collects images to be fed into the classification mode and a physical interface that can successfully indicate the result of the classification.

As for what was not implemented, the addition of these elements would have greatly improved the system, especially the inclusion of the MQTT interface. This would have allowed the user to monitor the device readings from anywhere and let them customise the bounds for what was considered safe for the environment based upon the use case e.g food inside the refrigerator.

The cost to create this device is relatively high as some of the components are either highly specific or need to be high quality. The total price breakdown of each major component is below.

Total Cost of whole system implemented as one device - ~£22.28		
Environmental Monitor Total Cost - ~£16.59		
Component	Price Per Unit	URL
Arduino Uno	£11.19	https://www.aliexpress.com/item/1005008083479174.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555315011&albag=1300723448300773&slnk=&trgt=pla-2333094955014654&plac=&crea=81295274240342&netw

		=s&device=c&mtctp=e
PZEM-004T	£3.99	https://www.aliexpress.com/item/4000330631886.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=554851195&albag=1306220947490970&slnk=&trgt=pla-4585238373891532&plac=&crea=81638863712723&netw=s&device=c&mtctp=e
LED	~£0.07	https://www.aliexpress.com/item/1005003844969782.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555315011&albag=1300723448300773&slnk=&trgt=pla-2333094955014654&plac=&crea=81295274240342&netw=s&device=c&mtctp=e
DHT11	£0.77	https://www.aliexpress.com/item/1005006144273755.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555190543&albag=1304021950430056&slnk=&trgt=pla-4585100935181475&plac=&crea=81501428383613&netw=s&device=c&mtctp=e
3-5V Motor with Fan	~£0.45	https://www.aliexpress.com/item/1005007341886295.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=554851195&albag=1306220947490970&slnk=&trgt=pla-4585238373891532&plac=&crea=81638863712723&netw=s&device=c&mtctp=e
270 Ohm Resistor	~£0.05	https://www.ebay.co.uk/itm/235329149666?chn=ps&norover=1&itemid=235329149666&targetid=4585169654799

		837&device=c&mktype=&googleloc=&poi=&campaignid=412354547&mkgroupid=1305120599331881&rlsatarget=pla-4585169654799837&abcId=9300541&merchantid=87779
1N4007 Diode	~£0.02	https://www.aliexpress.com/item/1005008751273458.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555315011&albag=1300723448300773&slnk=&trgt=pla-2333094955014654&plac=&crea=81295274240342&netw=s&device=c&mtctp=e
2N2222 Transistor	~£0.07	www.aliexpress.com/item/1005005776372882.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=554851195&albag=1306220947490970&slnk=&trgt=pla-4585238373891532&plac=&crea=81638863712723&netw=s&device=c&mtctp=e
Fruit State Detector Total Cost - ~£16.88		
Arduino Uno	£11.19	https://www.aliexpress.com/item/1005008083479174.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555315011&albag=1300723448300773&slnk=&trgt=pla-2333094955014654&plac=&crea=81295274240342&netw=s&device=c&mtctp=e
LEDs	~£0.14	https://www.aliexpress.com/item/1005003844969782.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555315011&albag=130072344

		8300773&slnk=&trgt=pla-2333094955014654&plac=&crea=81295274240342&netw=s&device=c&mtctp=e
OV5642 Camera	£3.60	https://www.aliexpress.com/item/1005004386240739.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=554851195&albag=1306220947490970&slnk=&trgt=pla-4585238373891532&plac=&crea=81638863712723&netw=s&device=c&mtctp=e
ESP-01	£1.95	https://www.aliexpress.com/item/33040500358.html?src=bing&aff_short_key=UneMJZVf&aff_platform=true&isdl=y&albch=shopping&acnt=135095331&albcp=555315011&albag=1300723448300773&slnk=&trgt=pla-2333094955014654&plac=&crea=81295274240342&netw=s&device=c&mtctp=e

5. References

Bijoy, M. H. I., Tasnim, S. Z., Awsaf, S. A., & Hasan, M. Z. (2024). *A large-scale dataset for fruit classification: Insights into fresh, rotten, and formalin-mixed samples* (Version 1) [Data set]. https://doi.org/10.17632/xkbjx8959c.1

Mouser Electronics. (2012). *DHT11 Humidity & Temperature Sensor.* https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

Innovators Guru. (2019). *PZEM-004T V3.0 User Manual.* https://innovatorsguru.com/wp-content/uploads/2019/06/PZEM-004T-V3.0-Datasheet-User-Manual.pdf

Holley, D. (12 Mar. 2025). *Understanding the Amperage of Refrigerators: A Comprehensive Guide*. AppliancesFirst. appliancesfirst.com/how-many-amps-use-a-refrigerator/

Rubell, B. (2019, July 23). *MQTT in CircuitPython*. Adafruit Learning System. https://learn.adafruit.com/mqtt-in-circuitpython/connecting-to-the-adafruit-io-mqtt-broker

TensorFlow. (2019). *Keras | TensorFlow Core | TensorFlow*. TensorFlow. https://www.tensorflow.org/guide/keras

TensorFlow. (2025, January 13). *The Sequential model | TensorFlow Core*. TensorFlow. https://www.tensorflow.org/guide/keras/sequential_model

OpenCV. (2025, May 8). *OpenCV: OpenCV-Python Tutorials*. Docs.opencv.org. https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Ultralytics. (2023, November 12). *YOLOv8*. Docs.ultralytics.com. https://docs.ultralytics.com/models/yolov8/

Arduino. (2024). Firmata Library. Arduino.cc. https://docs.arduino.cc/retired/hacking/software/FirmataLibrary/

Adams, K. (2025, April 28). *The Perfect Balance: What Should the Humidity Be in a Refrigerator? - Appliance Update*. Appliance Update. https://applianceupdate.com/what-should-the-humidity-be-in-a-refrigerator/

Gallagher, P. (2025, May 1). *How to store food safely in the fridge*. Which? https://www.which.co.uk/reviews/fridges/article/how-to-store-food-safely-in-the-fridge-a05PO7F3s4m3

Ai-Thinker. (2017). *ESP-01/07/12 Series Modules User's Manual OEM/Integrators Installation Manual*. https://usermanual.wiki/Ai-Thinker-Technology/ESP01E-4071614.pdf

Graham, E. (2024, September 6). *Food Waste Statistics for 2024 - Waste Direct UK*. Waste Direct. https://wastedirect.co.uk/blog/food-waste-statistics/