Hardware Requirements to Simulate the Project

- Argon NEO **Raspberry Pi 4 Model** B Case (AR_NEO_RPi4_2Gig_32GigSD) - This kit contains Rapberry pi, heat sink, HDMI connector and SD card
- Arducam **Auto Focus Pi Camera**, Autofocus for Raspberry Pi Camera Module, Motorized Focus Lens, Software Precise Manual Focus, OV5647 5MP 1080P

Configuration Steps for hardware

Raspberry pi 4 Model:

- Setup the SD card and install Raspberry Pi operating system on it.
- Connect all the parts and on power on you will see the desktop.

Arducam Auto Focus Pi Camera:

- Assemble the acrylic case: https://www.arducam.com/docs/cameras-for-raspberry-pi/camera-case/case-installation-pictures/
- Install python Dependency libraries: sudo apt-get install python-opencv
- Download the python scripts for focus control:github.com/ArduCAM/RaspberryPi/
- Enable the I2C0 port: chmod +x enable_i2c_vc.sh
    - ./ enable_i2c_vc.sh
- A reboot is required after running this script.
- Run the examples: Motorized_Focus_Camera_Preview.py : Manual focus in preview mode. Use the keyboard up and down keys to see the focusing process.

Device Code

Code on Raspberry pi

Motion_detect_upload_image_to_s3.py: This function depicts the motion in front of autofocus raspberry pi camera module and capture the photo and upload the captured image on s3 bucket(“smartcartphoto”). It also connects this raspi device to aws iot core and publishes the path of images uploaded on s3 bucket to the topic iot/motionDetected

P3picam.py: Threshold determines how much a pixel has to change to be marked as change. This code captures the image number of total changed pixels are greater than Sensitivity.

Cloud Code

Issue with using lambda function of AWS : As with the captured images from raspberry pi we have to run barcode detection algorithm and machine learning model , we have to provide these python dependencies as well as tensor flow libraries to zip file in lambda function. There is a maximum size limit of the zip file that can be uploaded. But our requirement was greater than that. Hence created an EC2 instance as our backend server to process the images.

Code on EC2 instance used as a backend server

Main Function.py: This code reads the messages published by raspberry pi on the topic iot/motionDetected. The messages consist of paths in the s3 bucket where images captured by camera sensor are kept. Images are read from S3 (s3Read.py) and fed to barcode detection algorithm. If code is not -1 then it is a packaged item. If code is -1 then it goes for fruit and vegetables classifier to determine the exact fruit or vegetable placed in the cart.

S3Read.py : It is called by Main_Function.py. It establishes the boto3 client connection to s3 bucket with the help of aws access key id and aws secret access key. After that it downloads the uploaded image from s3 bucket to this ec2 instance.

Barcode Detection

Barc.py : With the help of scikit-image library in python image is read as numpy array and if it is in RGB colour model, it is converted into gray scale by rgb2gray function. With zbar python library Scanner object is instantiated and image as a numpy array is provided as an input to the scanner object. The obtained result is decoded in ascii format to read it.

Python dependencies to be installed for compiling Barc.py

Machine learning model for fruit Classification

Dataset Used For training the model: Fruits 360 dataset: It contains 90483 images of 131 fruits and vegetables. For this project I have considered in total 10 classes of fruits and vegetables.

Labels = [‘None’, ‘Apple Red’, ‘Banana’, ’Kiwi’, ’Lemon’, ’Mango’, ’Onion White’, ’Pepper Green’ ,’Tomato’, ’Watermelon’]

trainmodel.py , predict.py

For Building the keras model, first tensor flow dependencies are imported. Training and testing data set path is provided. Input images are 100 by 100 pixels, but they are resized to 25 by 25 pixels to speed up the training process. Multi-layer convolution neural network model is trained to evaluate the features of the user image. It contains an input layer, some convolution layers, ReLu layers , some dense layers and an output layer.

Compiling CNN: Compile function is used which has three parameters Optimizer, loss function and performance metrics

Fitting the CNN: Fit method is called which takes first parameter as the training set with features. The second parameter is the *target* that you’re making the predictions on. The *batch size* represents the number of samples that will go through the neural network at each training round. *Epochs* represent the number of times that the dataset will be passed via the neural network.

Model is saved for later use.

Database Used: Dynamo Db

- Created two tables :

- all_items : which stores information about all the items in the mall . [barcode_no, item_name,item_price,description]
- user_carts : which stores the information of the items that are kept in user cart

Dynamodb.py

This file contains function put_user_cart .If the user puts fruits and vegetables in the cart , get_price_with_name function is called to determine the price of fruit from all_items database.

If the packaged item is kept in the cart, get_price_with_barcode function is called to query the price of the item from all_items database.

If the item is already existing in the cart then its quantity is increased in the user_carts dataset.

Else entry is made into the user_carts with quantity 1.

Android Application Code

The android application is listening for changes in the DynamoDB and displaying them on the activity and also displaying a payment pipeline.

Software Required: Android Studio

Emulator: Samsung M21 phone with developer options enabled

MainActivity.java:

This starts the application and ask for cart number. After taking the cart number it invokes the Bill activity.

Bill.java:

This the activity which holds the ItemList adapater for the items in the cart. It invokes the ask job of DBPoll.java. And passes the current activity

DBPoll.java:

This file is responsible for an async task which polls the dynamoDb for the items in the user cart after ever 10 seconds and updates the Item List held by bill and calls notify.

This calls the basic credentials to get access to the dynamoDb.

ItemsList.java

This is the ArrayList Adapter which refreshes the list from the DBpoll using the notify

MakePayment.java

This is invoked from Bill.java and contains various payment options. Currently these are only placeholder

Created 3 major layouts for the main , itemview and Make Payment activity.

Evaluation considerations for three different modules:

- Time taken from Device to cloud
  - Uploading Image from raspberry pi to s3 bucket: Approximately 200 images were sent from device to aws cloud. Number of exceptions that failed to upload on s3 bucket.
  - Sending path messages from raspberry pi to backend server (ec2 instance).
- Time taken on cloud
  - Time taken for barcode detection of 100 images.
  - For curved surfaces like cans, barcode detection algorithm does not work.
  - Training the machine learning model for the training data set consisting of 10 fruits and vegetables in total. Accuracy of the machine learning model.
  - Time taken to predict the type of fruits and vegetables for 60 images.

- Time taken from cloud to android application
  - Polling frequency to query the database for the item kept in the user's cart is 1 minute.
  - Updating the cart display on the android application with each update in the database "user_carts"