INTELLIGENT INVENTORY MANAGEMENT SYSTEM

(Project Report)

1. Team Member(s)

Shivam Khandelwal

2. Abstract

Today Internet of Things is driving the innovation very rapidly, encompassing every field gradually. This prototype is an ideology that can allow a plethora of devices to be connected and share huge amount of data to and from the cloud. Cloud based inventory management is a new focus in this Internet of Things era. A combination of smart applications, API's, internet (cloud), sensors and other technologies can restructure the supply chain and stock management process to become more responsive, streamlined, intelligent and real-time.

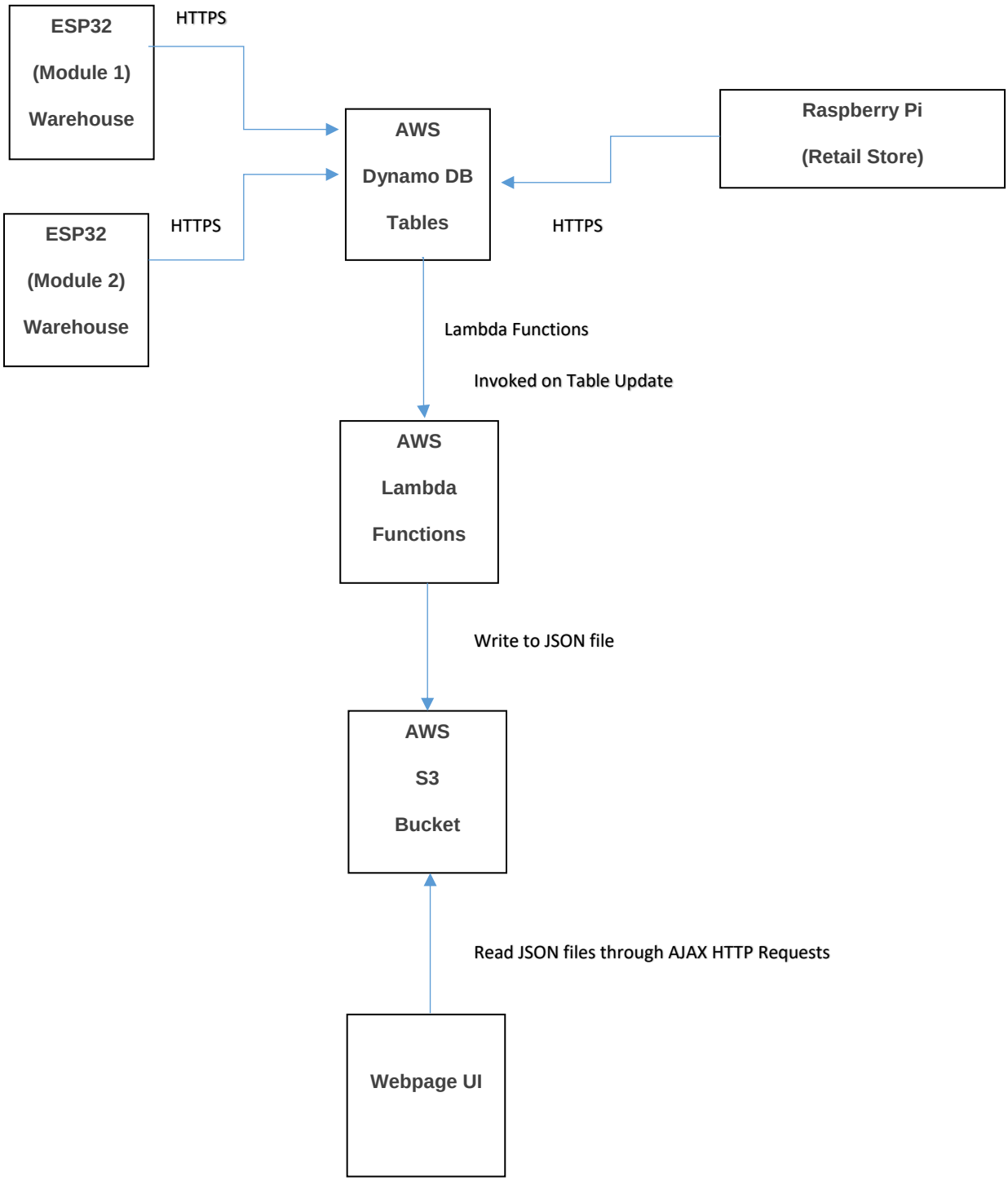
Warehouses and retail stores maintain large and diverse stockpiles which makes the stock management at different levels of the hierarchy - a complex process. Lacking proper planning and appropriate analysis of consumption pattern, entails disproportionate distribution of stock at retail store and warehouse. Eventually, it becomes a tedious task to observe regular sales, refill interval and stock required to cater guest visits.

3. Objective

In this project, we intend to develop a prototype to incorporate the following features -

1. Tentative purchase list generation and alert notifications to indicate stock required at warehouse.
2. Planning stock purchase based on stock availability.
3. Stock entry at warehouse.
4. Updating stock status, stock entry/receipt at warehouse on cloud.
5. Dynamic stock request monitoring at retail store.
6. Updating stock status and requirement at retail store on cloud.
7. Retail store inventory overstocking and under- stocking reduction.

4. System Diagram –



5. Project Description

Stock is generally maintained in two ways –

a. Packaged / Packed goods (pieces) - Entity
b. Loose quantity (by weight/volume/size) - Quantity

In this project, stock maintained as an entity is assumed to be bought and entered in stock by reading their RF tags. On the other hand, stock maintained in quantity will be tracked and entered by measuring weight.

Weight will be measured using strain gauge load cells (shown below) –

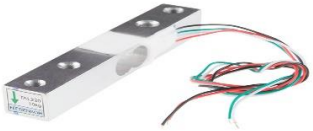


Image Source - https://www.sparkfun.com/products/13329

Packed goods will be tracked by reading their RF tags using the mrfc522 reader module -

Image source - http://www.hotmcu.com/mifare-1356mhz-rc522-rfid-card-reader-module-p-84.html

The information pertaining to the quantity of stock is uniquely recognized and regularly updated on cloud to be viewed by the manager (at retail store) as well as at the warehouse. Thus, standardizing and streamlining the logistic movement and inventory management at both warehouse and retail store.

WAREHOUSE:

1.	**Stock entry and updating initial stock status on cloud database.**
2.	**Updating real-time stock and requirement (if any) on cloud database.**

Whenever an item gets added to the inventory, the weight in the current state would be greater than the previous state's weight. On the other hand, when an item is removed, the weight of current state would be less than the previous state. The load cell accurately determines the quantity of items either consumed or remaining inside each container by measuring the weight. The manager (at retail store) will be able to set the threshold for every item. If it is not set, then the system will take the last refill quantity as the

maximum quantity. A percentage of which will be treated as threshold after monitoring one week's consumption.

A similar procedure is used to track packed items by reading their RF tags using RF reader module.

RETAIL STORE:

1.	**Updating real-time stock status on cloud database**
2.	**Updating real-time stock requirement (if any) on cloud database.**
3.	**Updating stock arrival / stacking on cloud database.**
4.	**Updating stock sale on cloud database.**

At the retail store, items will be added into the stock as they were added at warehouse. But instead of tracking stock of each item in real-time, stock status will be determined from sale of items and then calculating the remaining stock.

MANAGER (AT RETAIL STORE):

1.	**Checking real-time stock status of any item at warehouse or retail store.**
2.	**Fetching warehouse stock requirement from cloud database (and/or get notified).**
3.	**Fetching retail store stock requirement from cloud database (and/or get notified).**
4.	**Placing stock purchase orders for warehouse.**
5.	**Putting stock items on hold (to be purchased later) and get notified if there is an urgent need.**
6.	**Getting notified about purchased stock arrival and entry at warehouse.**

Apart from stock tracking, sale determination and stock entry at warehouse and retail store, the manager (at retail store) will be regularly updated about stock status and availability at warehouse. A tentative list will be generated for items below a certain quantity in the warehouse stock so that stock purchase can be planned accordingly. These items can be put on hold while tracking their real-time status at both warehouse and retail store. In case these stock items are near to being exhausted, the manager will be notified for placing immediate purchase order.

6. Project Implementation

Hardware Setup -

Warehouse -

1. ESP32 (Wi-Fi + BLE module) – for real-time stock status of each item.

ESP32 is a Wi-Fi IoT module which can be directly used the local Wi-Fi network to access the internet. Load cell and RF reader module will be attached to each of the two ESP modules which will act as independent nodes and push data on cloud using HTTP

Retail Store –

1. Raspberry Pi 2 – for store sales, stock monitoring by manager and for placing / planning stock purchase orders.

The Raspberry Pi will access the AWS using IAM credentials and .

Software Platforms /SDK's:

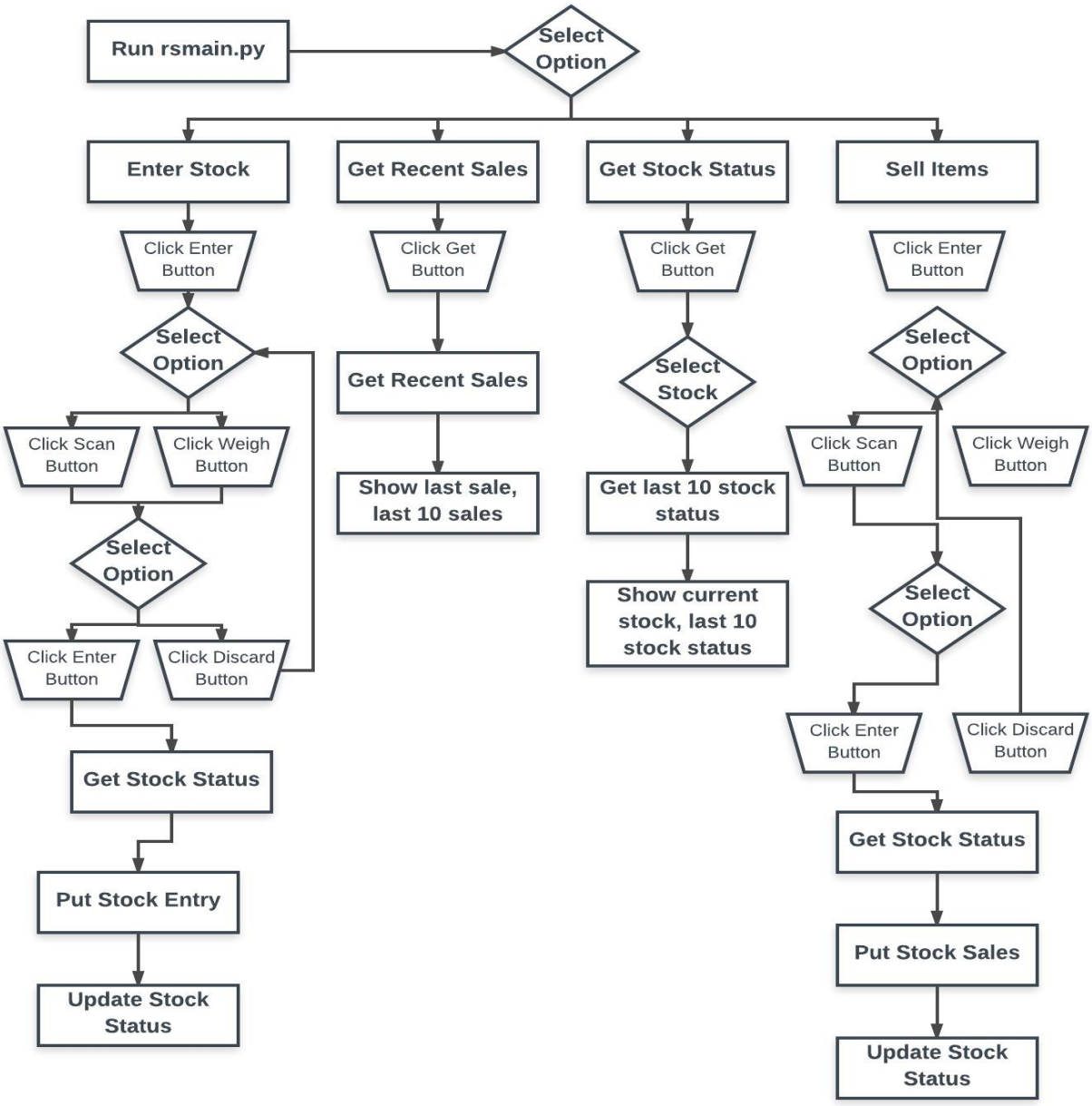
The raspberry Pi will connect to cloud (AWS IoT, Dymano DB, Lambda) using the AWS python SDK.

The ES32 is connected to cloud by –

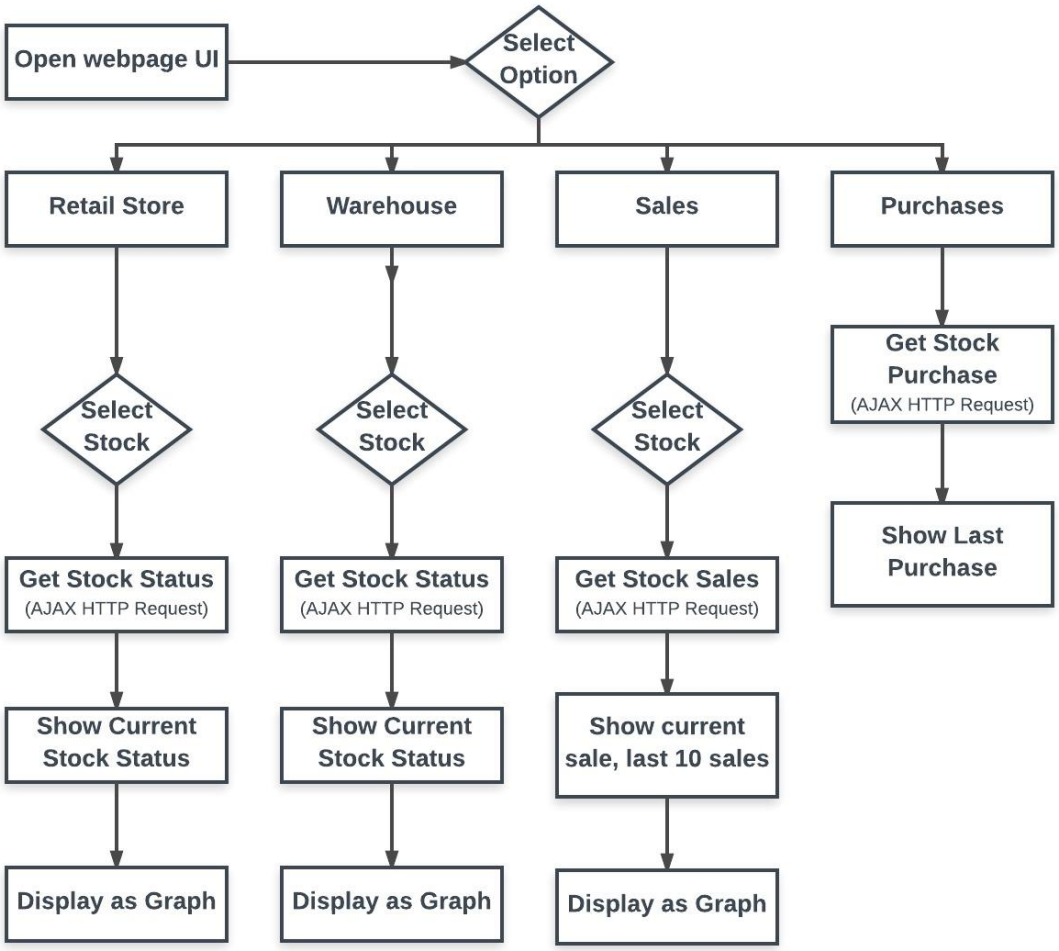
1. Programming the module in Arduino IDE (an HTTP connection is established)

7. Use Cases

At Retail Store –



Webpage UI –


Open webpage UI
Select Option
Retail Store
Warehouse
Sales
Purchases
Select Stock
Select Stock
Select Stock
Get Stock Purchase
(AJAX HTTP Request)
Get Stock Status
(AJAX HTTP Request)
Get Stock Status
(AJAX HTTP Request)
Get Stock Sales
(AJAX HTTP Request)
Show Last Purchase
Show Current Stock Status
Show Current Stock Status
Show current sale, last 10 sales
Display as Graph
Display as Graph
Display as Graph