CS-684-2018 Final Report

Automated Retail Store

Team Members

1. Deep Modh (140050002)
2. Neeladrishekhar Kanjilal (140050007)
3. Anuja Parab (17V972042)

Table of Contents

1. Introduction

***Computer Vision** based partially-automated store where customers are able to purchase products without using a cashier or checkout station.*

*The aim of the project is to *reduce computational cost* inherent in using Computer Vision on such a large scale.*

*We present **Embedded Systems** based solution to replace the 'checkout station' for convenience of customer by saving time. We plan to use location tracking along with weighing mechanism to identify purchased products.*

Our project, once scaled, has potential to impact millions of people's retail store experience.

2. Problem Statement

The idea is to make a store where customers can purchase products without using a cashier or checkout station.
The customers should never have to wait in a line.
Just Walk Out shopping experience

Knowledge of WHAT the customer is buying. Real-time tracking of weights of the racks in a shelf where the products are placed.
Knowledge of WHERE the customer is buying. Real-time positional tracking of customers, to get the shelf of interest.
Knowledge of HOW the customer is buying. A mobile application that will keep track of the actions and charge accordingly on exit.

3. Requirements

To obtain required result we primarily need two key features :
1. Weight tracking methodology which can also connect to IoT and notify whenever there is a change in th eweight of a rack/shelve.
2. Customer position tracking methodology where we need to know where the customer is at any given time to know whoch customer picked a product

For obtain this following are the basic functional amd non-functional requirements you must have for atleast making a satisfactory working demo to convince the partial idea and its ability to scale to size and circumstances

3.1 Functional Requirements

Digital Load Cell Weight Sensor

Hx711 Weighing Sensors ADC module

Esp32 Wifi Bluetooth Development Board

Mobile device with a GPS tracker/Wifi fingerprinting

3.2 Non-Functional Requirements

AWS IoT Services

Android Studio SDK

Arduino IDE(with libraries for HX-711 and esp-32)

3.3 Harwdare Requirements

Digital Load Cell Weight Sensor

Hx711 Weighing Sensors ADC module

Esp32 Wifi Bluetooth Development Board

Mobile device with a GPS tracker/Wifi fingerprinting

3.4 Software Requirements

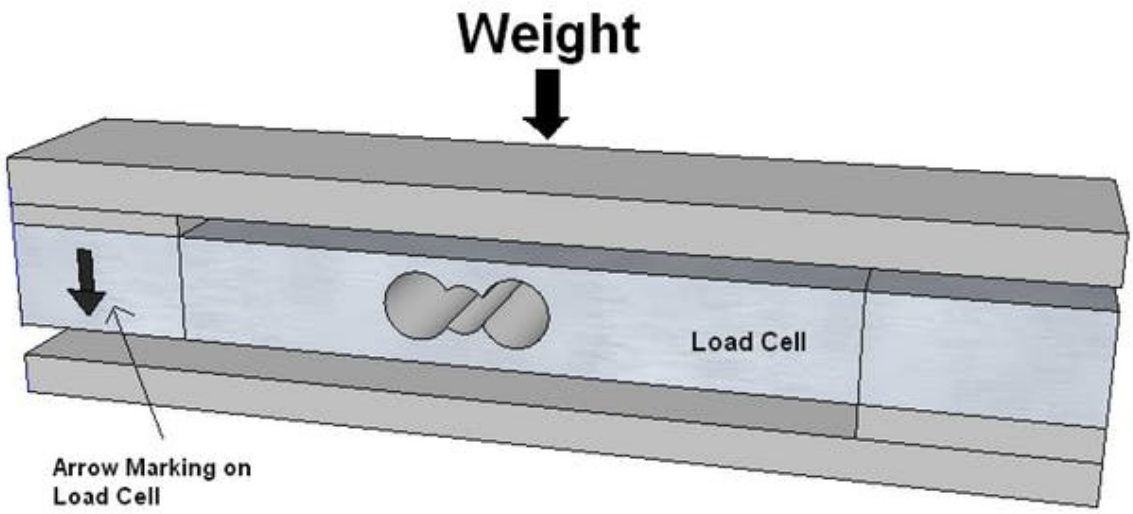
AWS IoT Services

Android Studio SDK

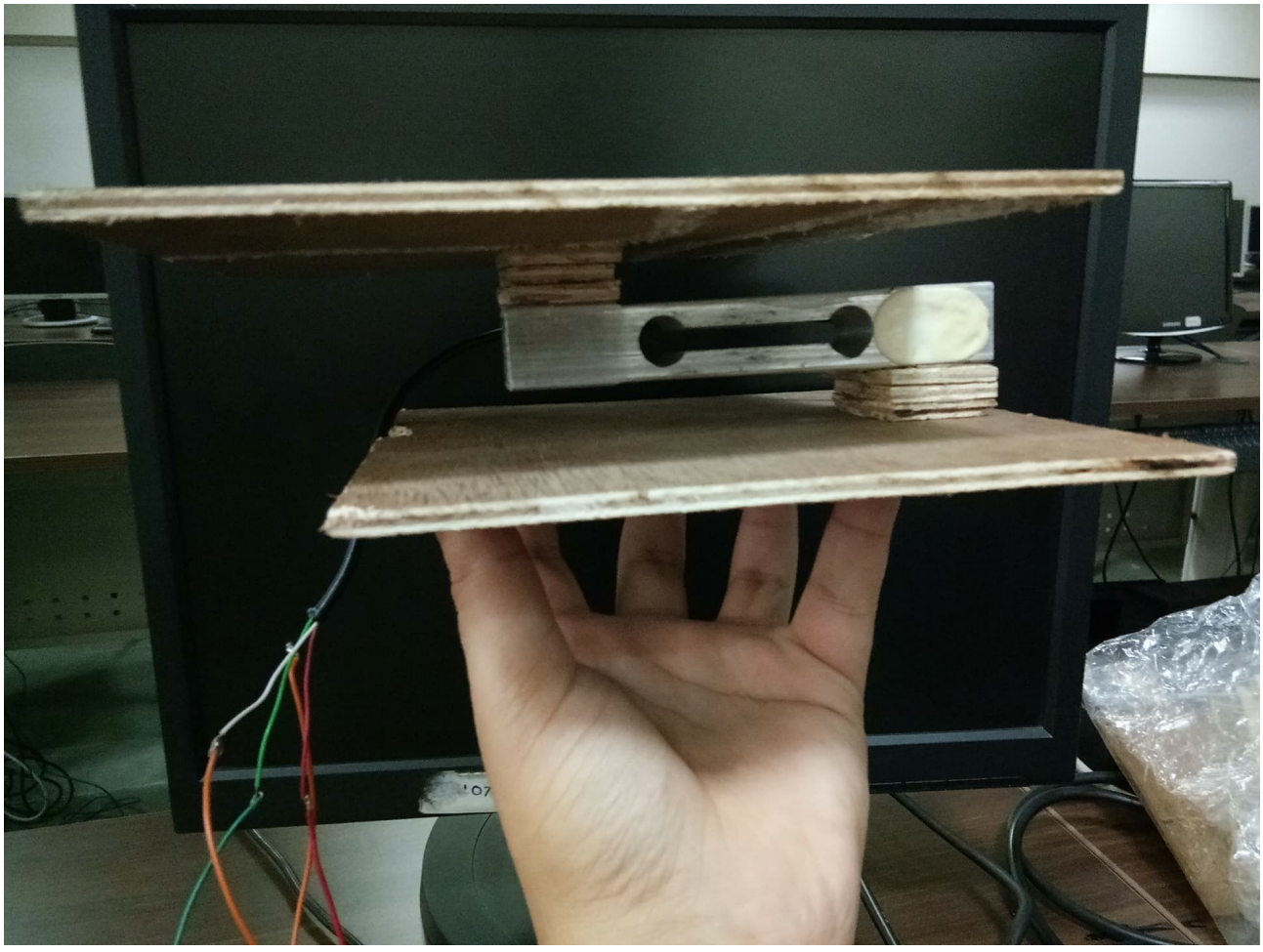
Arduino IDE(with libraries for HX-711 and esp-32)

4. System Design

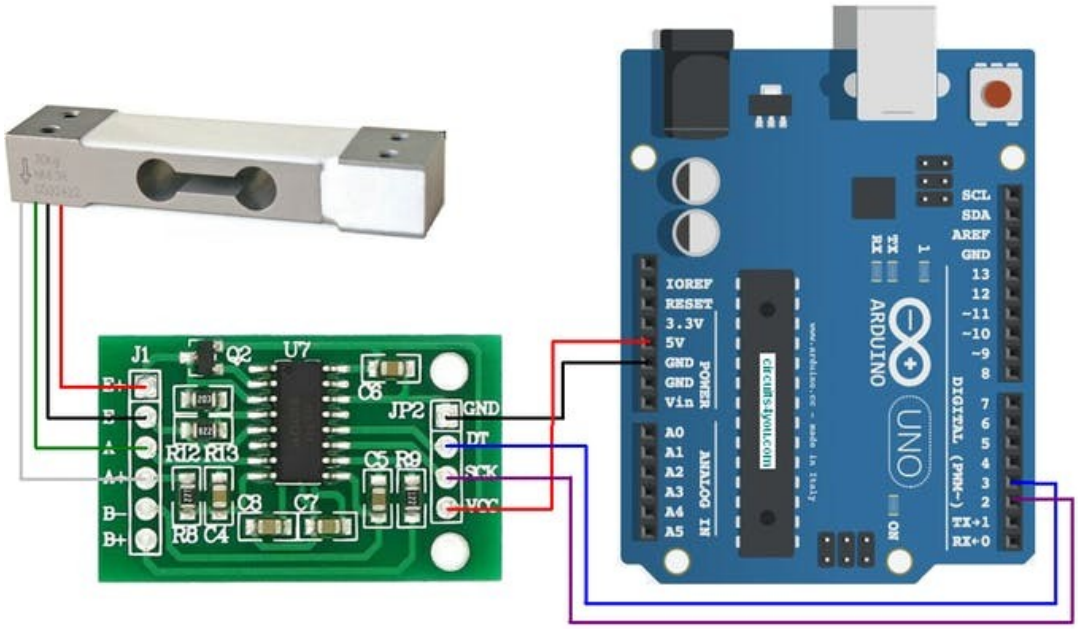
Load Cell mounting

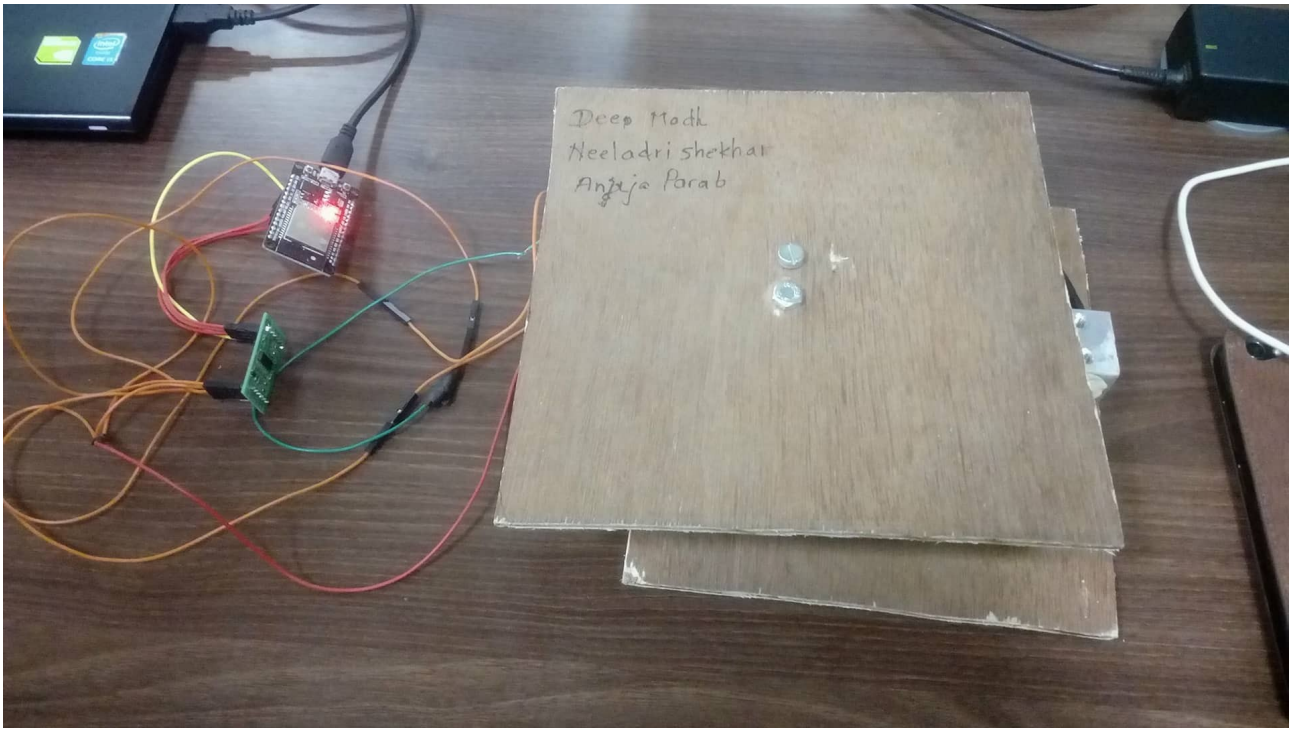


Our setup for the load cell :

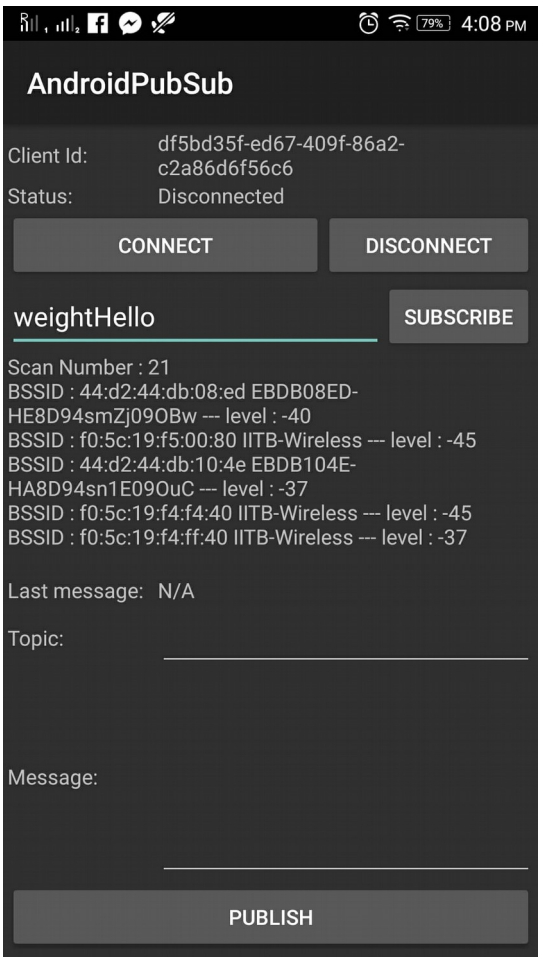


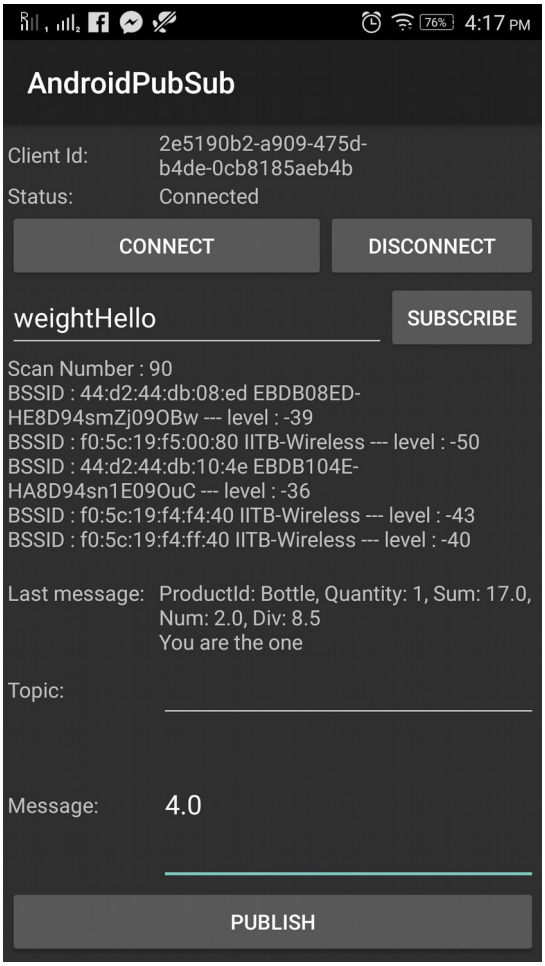
Next step for connecting hx711 to load cell and all that to you micro controller(esp-32)





Next step is to install the AndroidPubSub app on an android device. You should expect the following starting screen.





The new screen on weight change on the load cell is as shown in the second image above

5. Working of the System and Test results

The Digital Load Cell Weight Sensor will give Analog signals corresponding to the weights.
This is then transferred to the Hx711 Weighing Sensors ADC module for getting a Digital value of the weights.
These values will be uploaded to Cloud using the Esp32 wifi module(Relying on the AWS IoT services at the backend)

Primarily we had planned to rely on the inbuilt GPS of the mobile device which have an accuracy of less than half a meter. On a terrace/balcony based store this method of tracking works really well.
However for indoors(within walls), because of attenuation and scattering, the accuracy of GPS may drop significantly. Solutions to these would be using beacon tracking based upon Bluetooth, Wifi or RFID.
The prototype uses the in-built wifi module of esp-32 and the consumer mobile for localizing.

```
Reading: 0.87 kg
Reading: 0.87 kg
Reading: 0.87 kg
Reading: -0.00 kg
picked something
Publish Message:{'messageType':'weightChange','productID':'Bottle','quantity':1,'wifiSignal':[{'BSSID':'BE:2F:3D:89:EC:60','strength':-37},{'BSSID':'44:D2:44:DB:10:4E','strength':-57}]}
Reading: -0.00 kg
Received Message:{'messageType':'weightChange','productID':'Bottle','quantity':1,'wifiSignal':[{'BSSID':'BE:2F:3D:89:EC:60','strength':-37},{'BSSID':'44:D2:44:DB:10:4E','strength':-57}]}
Reading: -0.00 kg
Received Message:{'messageType':'purchase','customer id':'Neeladri', 'product Id' : 'Bottle', 'quantity':1}
Reading: -0.00 kg
Reading: -0.00 kg
Reading: -0.00 kg
```

The above result from the demo is a screen shot of the Arduino IDE serial when a weight is picked from the load cell and there was a customer in its close proximity.
The message type ‘purchase’ is published when a customer has picked the product and the details of that customer are now known.

6. Discussion of System

a) *We planned to have specialized shelves where each of its rack would have its own weight tracking mechanism. However we coudn't manually finish one.*

b) *The mobile application actually keeps a track of the products which a customer picked and shows it accordingly. This method can be scalled a lot to multiple customers using the same code.*

7. Future Work

Every component is reusable. Infact we can expand to more specific hardware for dedicated tracking methods.

1. esp-32 can be replaced by an esp-8266 which are more cheaper. We used esp-32 keeping it as a back up option if wifi tracking fails, we could work the Bluetooth tracking, which has a much more reduced range but higher accuracy.

2. Out door locations do not need Beakons/Routers as GPS alone is self standing for the purposes.

3. The android application is a very basic debug app, and can be modified with User experience and customer satisfactio in mind.

4. Implementation of a Database at the MQTT servers(AWS DynamoDB) for customer tracking and product management.

5. Cognito based user identification, implementing roles via IAM policies for multiple User-Pools.

8. Conclusions

Probably a 'Vending Machine' is the closest working model towards pick and leave shopping convinience. Amazon Go uses all what we have done, built upon a bluetooth tracking method. They also have camera inputs to further confirm the customer – product relation.

9. References

Python for Serial I/O : http://docs.python-guide.org/en/latest/starting/install3/linux/

Arduino for esp32 cod : https://www.arduino.cc/en/Guide/ArduinoUno

Espresso libraries sp : https://github.com/espressif/arduino-esp32

Amplifiers of espress : https://www.hackster.io/MOHAN_CHANDALURU/hx711-load-cell-amplifier-interface-with-arduino-fa47f3

AWS setup in esp32 ad : https://github.com/ExploreEmbedded/Hornbill-Examples/tree/master/arduino-esp32/AWS_IOT

JAVA for Android stdo : https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html

Android Studio for UI : https://developer.android.com/studio/install

AWS setup for Android : https://github.com/awslabs/aws-sdk-android-samples/tree/master/AndroidPubSub