Barcode Item Scanner System(BISS) with Inventory Logging

Samartha K Venkatramana, Nikhil Swamy

***Abstract*—An attempt to use the Raspberry Pi with attached camera as a barcode scanner connected to a inventory logging system. The use of mosquitto MQTT provides greater security, while sending client data to the server.**

***Index Terms*—RaspberryPi, Barcode, Computer Science, Programming**

I. Introduction

Raspberry pi emerged around 2011, and has gained wide popularity for its internet based applications, such as Internet of Things, etc. The applications of Pi are always expanding and ever engaging scientists and engineers to apply their minds on these topics to generate something unique. Over the last five years, raspberry pi has grown rapidly, to support these wide variety of applications (Ebton, 2013). In most warehouses, stores and such outlets, the computers are connected to the company server through standard internet protocols. Exploring in this domain, we figured that safety can be compromised and in some cases, it is not even feasible to maintain or aid inventory logging through this. The security aspect is by hacking and changing inventory values, the intruder might cost the company more than necessary, also, in large companies, stock control becomes harder and harder, hence they largely depend on accurate computer data. If this data is manipulated, the company requires a fixed amount of time to detect these intrusions. Hence, we believe that by using custom certificates and double or triple layer of encryption, protection and other such methods, we can scale up the security (Blake, Simon & Alfred). Various endpoint protocols such as SSL & TSL provide a wide range of security options to the developer. These protocols joined with hashing can provide a great deal of security. But since the protocol has caching of passwords and certificates, the authentication process involves storing of data and an intruder can switch certificates. Instead of depending on the standard system of authentication, we can encrypt the data and use our own certificates, which might provide better security. Owing to the fact that raspberry pi has a lot of libraries and features, we can exploit some of them to provide a higher level of safety.

The Internet of Things (IoT) is the network of physical objects, devices, vehicles, buildings and other items embedded with electronics, software, sensors, and network connectivity that enable these objects to collect and exchange data. The IoT allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit.

II. Related Work

"IOT Based Stock Verification System Using Raspberry PI, Barcode Scanner and Android Application." by Danawade, Vishal, et al. is a paper published by the students of a reputed engineering college in India. They talk about the use of raspberry pi for stock verification, which has been used widely in places such as libraries. Their project talks about how they switch the system, instead of carrying books to the fixed location computer, they claim that the user can just carry with him a raspberry pi with a barcode scanner to scan the items. He need not do any extra work carrying the books back and forth. This system allows for updation of stock details quickly. Moreover, they connect the Pi to an android device, thereby increasing the ease of access.

In our project, we utilize a similar approach, where in we use the portability and modularity of the pi to help end point employees at various storehouses or warehouses to quickly and efficiently scan the stocks while auditing. This system is similar but not the same as the technology we are using is based on IOT MQTT security, which the former Stock Verification approach by Danawade doesn't include. We also talk of the possibility for household usage. Most medication comes with expiry dates embedded in the barcode. Even by just making notes of the expiry dates for other commodities, we can develop a system which warns us just before the product goes bad. This system will help us judiciously use our food and medications, with little or no chance of wastage.

III. BISS - Barcode Item Scanner System

A common problem seen in any household is not knowing what all items are currently present in the house, if they are perishable and if so then what is there expiry date. If there was a system that would let the user scan the item after purchase and store the data about the item, by alerting the user at the time of its expiry, we can curb the wastage of food and other such perishables. This project aims to provide a workable solution to make life easier and keep things more organized and manageable. Another application of our system involves using this data over hand-held devices or at the company server. Our system manages both types of data, but we have only implemented it partially. Anyone can easily create a barcode scanner using the raspberry pi, and I will not delve into those details for now. We will basically concentrate on the security aspect and using MQTT client server architecture to safely transmit the data across various devices.

A. Proposed System:

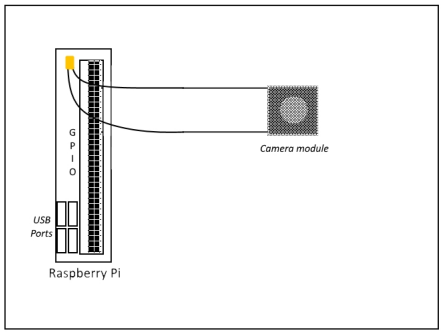
1) System components: We have used the following items in our system:

- Raspberry Pi
- Wifi Modem/Router
- Camera Module

2) Software Components: On the software ends, we require these factors:

- Python
- Raspbian
- zBar python lib
- paho mqtt

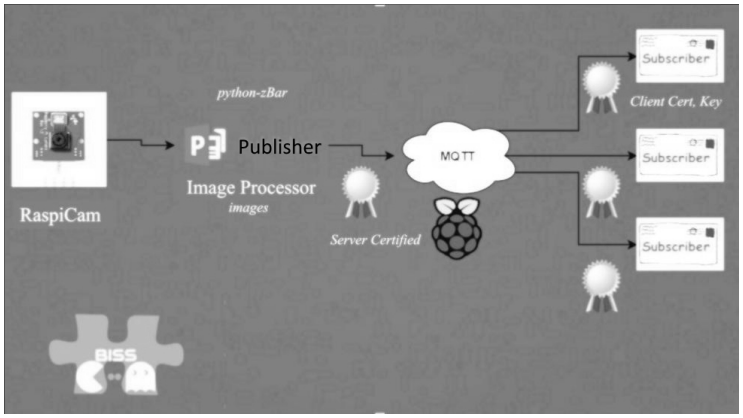
3) System Architecture: The following diagram represents the basic connections required to use this system. Most of the processing is done server side so on the client side, ie. raspberry pi client, we just take inputs and do minor processing.



The system involves connecting the pi to a camera module, and using the camera slot. Then we scan the barcode, using built-in capture commands. Next, we pass the image file as parameter to the python program, which converts the image to a barcode. This is easy to accomplish using the zBar library. // The next stage is to use the MQTT server. We installed the server and dependencies using python pip. We know that MQTT offers username/password based authentication, along with pre-shared key, for encryption and decryption of transmitting data.

4) Steps involved with MQTT:

1) Luckily, MQTT is easy to use and setup. But before that, we need to generate or grab our certificate and key files, along with server certificate.
2) Generate the certificate and key file using the repository package signing key. There are many github repos containing generators. Using one of them, we generate the key and cert pairs, etc.
3) Next we modify the *mosquitto.conf* files to update the certificate and other details.
4) At this point, we need to restart the server.
5) Next, we need to install paho mqtt, a python module to implement our code on both ends.
6) Then, we changed anonymous to false, to stop anonymous logins. We created a username and password for login through the pwFile.
7) Then we subscribe to a topic by running our subscriber.py
8) Later, we publish to the topic from the image processor so that we can send the necessary data over to the server.
9) We send the barcode, and date as an example.
10) We created a flag on the subscriber end which marks the end of client side publish, namely, "EOInput".
11) Upon reception of this message from client, the server tabulates all the data and outputs the table.



This image shows the setup.

IV. Analysis

We see that the system is successfully implemented, given the setup, the usage and modeling, anyone can achieve it. The following points encapsulate the analysis of the system:

- The images served to the python program on client side must be clear, concise and devoid of anything other than the barcode. We can call such an image a clear barcode.
- If the zbar program fails to identify the barcode in the image, then we will have to try again with a fresh image or change the library itself.
- Portability of the product largely depends on the fact that every network has unique network IP address, which means we need to generate the certificates and keys for each network the system intends to work with.
- Its easy to damage the camera module by using wrong power supply with the raspberry pi. Camera is a critical component of this project.
- Using the pi camera is not a good idea as the images captured aren't clear enough for the zBar library to decode the barcode.

V. Evaluation

The performance of a typical raspberry pi is categorized less than most of our hand-held devices. Hence we must keep the processing power in mind while writing applications for it. Having kept these in mind, we conclude that minimizing the processing on the raspberry pi side is the only option. Hence our program does only three specific tasks. Capture

the image and produce the barcode and then publish it to the server.
BISS succeeds in sending the data and generating tables. These values can later be exported.

VI. Conclusion

After working extensively on this project, we conclude that this system is clear success and will help future warehouse managers to log their inventories in an efficient manner. The system is very easy to setup and easier to use. The data sent can be encrypted in a number of ways to provide even better security. Since the assembly of the system doesn't require much expertise, this can be used by home users and employees alike. The system for a home user is fundamentally similar and should not take much time. MQTT has proved be highly secure and can do wonderful things in the field of IoT. Lastly, we witness that the system is capable of completing the said task and hence we were successful in our endeavor.

References

- Severance, Charles. "Eben Upton: Raspberry pi." Computer 46.10 (2013): 14-16.
- Blake-Wilson, Simon, and Alfred Menezes. "Authenticated Diffe-Hellman key agreement protocols." International Workshop on Selected Areas in Cryptography. Springer Berlin Heidelberg, 1998.
- Danawade, Vishal, et al. "IOT Based Stock Verification System Using Raspberry PI, Barcode Scanner and Android Application." International Journal of Engineering Science 6361 (2016).
- Mosquitto, An Open Source MQTT v3.1/v3.1.1 Broker, Documentation, 2016
- Python Software Foundation, paho-mqtt 1.2, 2016
- mosquitto.conf the configuration file for mosquitto, 2016