Envisioning Hong Kong with E-Wallet Octopus System using NFC

Author: Jennifer Liu

On June 29, 2007, Apple announced its for first ever iPhone. That was when phones were no longer just means for messaging, texting, checking emails or calling. With the release of the first ever successful smartphone buy out, it could install games, applications written by third parties and so forth.

Fast forward to 2014, Apple and Google both released the Apple pay and Android Pay. It took a while for the market to get a hold of this newly developed payment due to security concerns by customers and installations frustrations. However, as consumers slowly eased into the technology, phone payments became increasingly popular, especially in China.

Due to its popularity, I wanted to allow Hong Kong people to have the option to take the subway, take the bus, pay for 7-eleven snacks without having to bring an Octopus Card. I have created a prototype demonstrating that this is completely possible. This prototype is by no means ready for production. I acknowledge that it has many flaws, but this is also why I wanted an industry leader like Octopus to seriously criticize its flaws and suggest improvements for it.

A video is available on the site, and to further understand this, it is advise that you watch before continue reading.

The following whitepaper will be a more detailed description of how to use it, and implement it. The paper will be broken into 2 sections.

Section 1: How to Use
This part of the paper will discuss how Octopus Pay is used from a consumer's perspective.

Section 2: Logistics, Cost, other necessary technological concerns

- *How to Implement:* Logistics on Implementing and expected completion time
- *Hardware Cost Breakdown:* Hardware needed to build this and expected cost
- *Restrictions:* Discusses technological restrictions using this device
- *Technological Exceptions/State Checking:* Building a robust, resilient system by ensuring all possible outcome are checked and handled

Section 1
How to Use

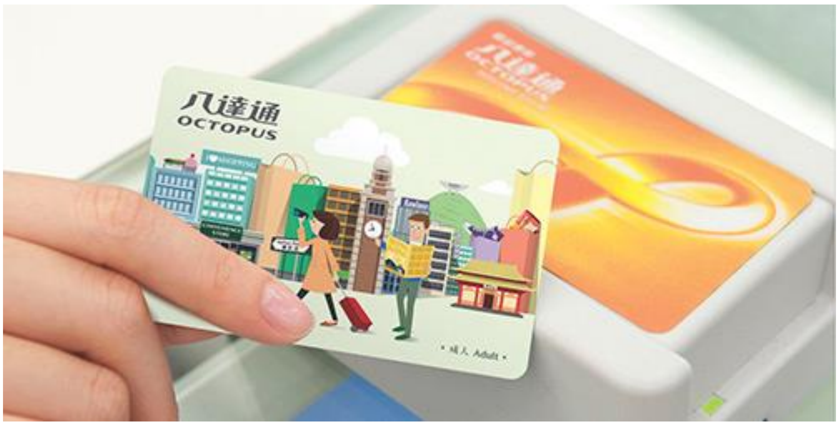

Figure 1 – Existing Octopus Receiver

Once again, what I describe here is an extremely rough and general idea. Implementation and usage will need a considerable amount of tweaking and adjustments.

The application is very simple to use. The hardware you saw at the beginning of the video is emulating the hardware that is hidden under the existing Octopus receiver (refer to Figure 1). Prior to using this technology, users must download an application that I have wrote. Once they tap their phones on the receiver, it will automatically open the application indicating transaction success or failure. User can then press the home button shown on screen to view their new transaction, or even top up their transaction. When they exit the station, they will tap their phone again, and payment will be processed and viewable from the application.

Section 2
How to Implement

To implement the software, we need to do the following:

1) Replace the hardware underneath the MTR booth – Depending on whether you have the NFC transistor, you could do without replacing
2) Integrate the hardware with the gate logic (permitting user to enter the MTR), and sound system to produce the beeping noise
3) Make modifications to current Octopus application
4) Integration, Security testing

Remarks

Suppose that hardware does require replacement. You do not have to replace every booth. Nonetheless, you do need to implement at least one booth in every station during the first wave of implementation. The reason you need this is because it needs to know which station the user started at and where they exited.

Additionally, the hardware is RFID/NFC capable. Thus, when you replace this hardware, users can choose between Octopus card or their phone.

I acknowledge that an Octopus Phone application has already been written. However, for me to make this technology work, I had to write a separate one as it required separate dataset and modification to existing software. If this proposal successfully goes through, all I will have to do is make modifications to the existing application.

I am also not a designer, so the application I have built is are bone design. It would be recommended to employ awesome designer to make this more appealing.

Hardware Cost Breakdown

Note: Hardware generally costs less when purchased in large quantities.

Hardware	**Cost (HKD)**	**Quantity**
Adafruit PN532 Breakout Board	$ 280.73	10-99
Arduino UNO Rev3	$171.75	1
Feather Stacking Headers – 12-pin and 16-pin female headers	$7.81	100+
Male-Male Jumper Wire (Which you guys probably already have)	$30.84	For so much that it will be sufficient to be 50 of these
Total	$491.13	

It costs only $491.13 HKD to build this prototype. I am assuming Octopus will be designing their own circuit boards and letting third parties manufacture it. In this case, it would be even cheaper to build the hardware.

Restrictions

Currently, only android phones are capable of this technology. Apple phones do not natively have the technology to do it.

Technological Exception/State Checking

One of the most important principles in building any technology is to check for all exceptions. If failures do occur and we did not catch it, it should exit the system gracefully regardless. In my demonstration, all the mentioned exceptions are checked. There could be more that I am not aware of.

There are only two final possible outcomes in this transaction, it can either be a failed transaction, or a successful transaction. It first checks whether the Octopus id inputted is correct, it then checks whether there is sufficient fund. If both criteria are satisfied, it is a successful transaction. Otherwise, it is a failed transaction. If there were other failures that were unknown to the program, it should return to the home screen or close the application. (This is marked as technological failure on figure 3.1). It would be best to demonstrate this in a state diagram, so we can capture every possible state for the most optimal user experience.

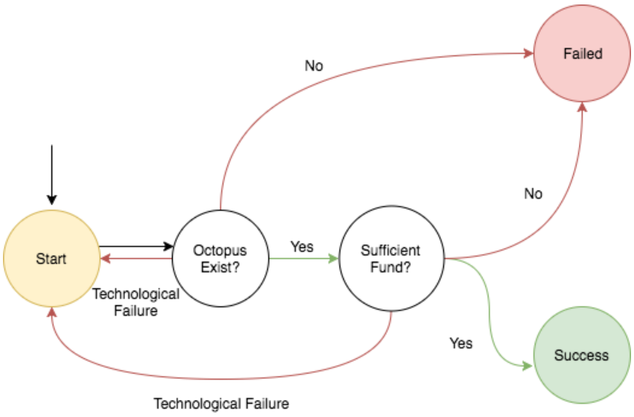

Figure 3.1 – MTR Application State Diagram

For demonstration, I simplified the logic slightly. When users tap on the phone, the payment is immediately processed. In reality, the payment should only be processed when a user completes a trip. By completing, I mean tap from entry station, and tap again from their exit station. A completion is when we finish the cycle in the state diagram. If user tries to exit with insufficient fund, it will prevent it from exiting, thus stuck on the exit state. Below is a diagram demonstrating where payment processing should occur.

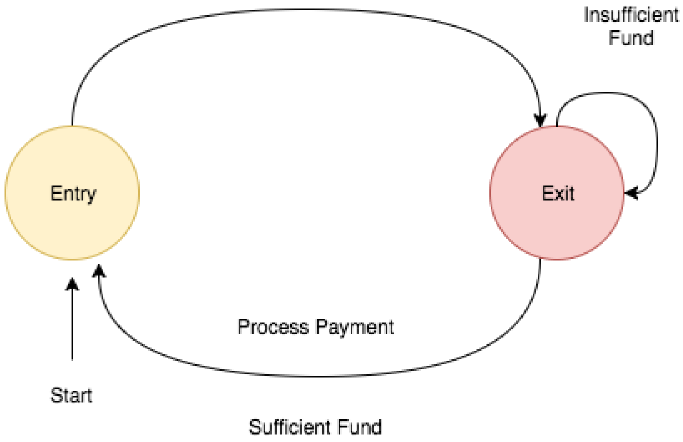

Figure 3.2 – MTR Transaction State Diagram