A Better Laundry System for HKU Dorms

fyp22023
Ho Joshua (3035689831)
Atiab Bin Zakaria (3035781230)
Outline

• Introduction
• Objectives
• Methodology
• Results
• Conclusion
Introduction

• Digitalization has become integral in people’s lives
  • E.g., Food ordering apps, mobile banking

• Laundry in HKU dorms is a manual process
  • E.g., Must physically go to see availability, the time remaining and able to use cash only

• Inconvenient and inefficient for students
  • E.g., Students living many floors from the laundry room waste a lot of time
Current Solutions

• Many universities around the world have a smart laundry system.
  • E.g., The Hang Seng University of Hong Kong have their own laundry app and most US universities have one as well.

• Allow students to do laundry tasks using the app
  • E.g., Check availability, time remaining, e-payment
Current Solutions

- CSC Service Works, a laundry solutions provider, provide most of the US colleges these smart laundry systems.
Current Solutions

• CSC Service Works has installed proprietary digital laundry technology on more than 15000 machines across 1000 academic institutions.

• CSC Service Works uses IoT technology and equips machines with Wi-Fi enabled devices to create a network of smart washers

• Network machines send data over to a secure cloud-based backend and finally to user applications.
Related Work

- Previous work on a smart laundry system have been done by other students

- Lacked certain elements:
  - Only gave a yes/no status for the washers
  - Lack of functions e.g., only status of washer is given
  - Did not leverage IoT devices
Objectives

• Build an app with the following functions:
  • Time checking
  • Washer reservation
  • Washer availability
  • E-payment
  • Notifications
  • Energy consumption dashboard

• Make dorm student’s lives easier and increase productivity
Methodology

• Next.js
  • React-based frontend framework

• FastAPI
  • Python-based API development framework

• Stripe
  • Secure e-payment gateway
Methodology

• Raspberry Pi Model B+
  • 1.4GHz 64-bit quad-core processor
  • 4 GB Memory
  • Wi-Fi and Ethernet port
  • Linux-based RaspberryOS

• Suitable for our use case
Methodology

• Raspberry Pi Model B+
  • Pre-process the raw capture of the display time
  • Turn on/off smart plugs of the washers at appropriate times
  • Process QR codes provided by students to allow use of washer
Methodology

• Pi Camera V2.1
  • Extract the time remaining of the washer for the web app
  • 8 megapixels and is 25 x 24 x 9 mm
    • Quality is more than sufficient for the use case
  • Mounted near the washer
Methodology

• WHD09 Smart Plug
  • Automatically switch on/off washer upon usage
  • Enforce the reservation mechanism
  • Record the energy used and display in the web app
Methodology

• Tuya IoT Platform
  • Cloud-based platform providing many kinds of services and resources for managing and controlling IoT devices from different brands
  • Benefits:
    • Has a data center located near Hong Kong thus no lag
    • Integrated with devices from many brands e.g., Samsung, Philips
    • Well-documented APIs and easy to use interface
      • Easy development process
    • Good security measures such as encryption and authentication i.e., Hacking to the smart devices is not easy
Methodology

• Tuya IoT Platform
  • Has useful cloud development features
    • Provides an asset-user system to manage and control devices
      • Can organize devices i.e., the smart plug in our case, to different hierarchies and groups
      • Leverage this development feature to split the smart plugs used into different dorms
Methodology
Methodology

• Tuya IoT Platform

  • Several APIs are used
    • /v1.0/iot-02/assets/{asset_id}/devices
    • /v1.0/iot-03/devices/{plug_id}/commands
    • ...

• Tuya provides a Python and JavaScript SDK to connect to the Tuya IoT platforms easily and call APIs
  • Using the CLIENT ID and CLIENT ACCESS
Methodology

Response

```
{
  "result": [
    {
      "id": "6c3e8b0f824770c4dfvcra",
      "status": [
        {
          "code": "switch_1",
          "value": true
        },
        {
          "code": "countdown_1",
          "value": 0
        },
        {
          "code": "add_ele",
          "value": 6
        },
        {
          "code": "cur_current",
          "value": 0
        },
        {
          "code": "cur_power",
          "value": 0
        },
        {
          "code": "cur_voltage",
          "value": 2131
        }
      ]
    }
  ]
}
```
Methodology

• SSOCR
  • Needed to extract time from the washer’s display
  • SSOCR (Seven Segment Optical Character Recognition) is used over TesseractOCR
    • TesseractOCR not trained on seven segment fonts by default
    • SSOCR specifically for seven segment fonts
  • SSOCR uses a deterministic algorithm i.e., not machine learning
    • Suitable for simple tasks like recognizing seven segment digits
    • However, image preprocessing needs to be done as image input have to be in a certain condition
      • E.g., Images must be monochrome.
      • Foreground pixels = pixels of the time to be extracted
Methodology

• SSOCR
  • Algorithm divided into two parts – Segmentation and Character Recognition
  • Segmentation
    • Finds different segments of an image that may be a digit
      • Finds the left border by finding the first column with a foreground pixel and right border by finding the first column with no foreground pixels
      • Similar process for the top and bottom border but will allow gaps since seven segment digits may have gaps in the middle section
Methodology

• SSOCR
  • Character Recognition
    • Based on the number of parts it contains (possibly 7 total parts in total).
    • Vertical scanline used in the middle to see if any foreground pixels are contained in the upper 1/3, middle 1/3, bottom 1/3 to find if the horizontal parts exist
    • Two horizontal scanlines a quarter from the top and bottom to check for the other 4 parts
    • A digit that has a width of less than one quarter of it's height is recognized as a one
Methodology
Methodology

• SSOCR
  • Many problems with using raw images.
    • SSOCR only works with monochrome images
    • Skewness will affect the result as the result is based on the position and orientation of the digit
    • Noise i.e., other non digit foreground pixels or shapes, will affect the result as they may be identified as digits themselves
    • Image pre-processing must be done.
    • SSOCR provides some commands but are not flexible or powerful enough
      • Need more control using OpenCV
Summary

- The FastAPI and Tuya cloud sends information to the web app and interacts with the hardware components
Results

• Image pre-processing
  • Similar with first presentation with minor and new changes
  • Three main steps
    • Changing to black and white
    • Removing noise
    • Fixing skewness
Results

• Image pre-processing (Monochrome process)
  • Mask the image
  • Find out the color of the time and turn all pixels with the same color to white and the rest black
  • Problem – What is the best way to represent the color of the time pixels?
    • Use an appropriate color space
    • Color space is essentially a representation of a color using numerical values
    • Two popular color spaces are RGB and HSV
Results

• Image pre-processing (Monochrome process)
  • RGB
    • Represent red, green and blue color intensity
    • Possible values for each range from 0 – 255
  • HSV
    • Stands for hue, saturation and value
    • Hue represents the color, saturation represents the intensity of the color and value represents the brightness
  • HSV is used over RGB because it is more intuitive, and the color is separated from the luminance
    • Easier and more accurate to represent the color of the time
Results

```python
hsv = cv.cvtColor(image, cv.COLOR_BGR2HSV)

# Define lower and upper limits of the number colour
number_color_lo = np.array([0, 10, 190])
number_color_hi = np.array([40, 200, 255])

# Mask image to only select colors in the range
mask = cv.inRange(hsv, number_color_lo, number_color_hi)

# Change those points to white
image[mask > 0] = (255, 255, 255)
image[mask <= 0] = (0, 0, 0)
```
Results

• Image pre-processing (Fixing skewness)
  • Idea: Find the angle of rotation of one of the digits and rotate the whole image by the same angle
  • The angle can be achieved by the minAreaRect function of OpenCV which returns the minimal sized rectangle enclosing the digit and its angle
Results

• Image pre-processing (Fixing skewness)
  • Problem 1: If apply a minimal area rectangle to everything in the image, which rectangle contains a digit?
  • Assumption: It is reasonable to assume the biggest min area rectangle contains a digit and gives the angle we need
    • The time usually makes up most of the space in the display
Results

• Image pre-processing (Fixing skewness)
  • Problem 2: The angle given may solve the skewness problem but the result after applying the angle makes it vertical
  • Solution: If the width of the minimal area rectangle of the digit is larger than the height, it means that after rotating it the orientation will be vertical.
    • All seven segment digits have a height > width
    • Depending on the slope of the minimal area rectangle, rotate it by -90 or 90 degrees

01:39
Result

- Image pre-processing (Noise removal)
  - Get all the regular bounding boxes of each parts in the image.
    - Assume the largest bounding box is a digit, since the time takes most of the space in the display
    - Compare the rest of the bounding boxes with it
    - Other digits should have the same area and height
    - Non digits should have a different area or height
    - No need to use the minimal area rectangle since the image is not skewed anymore
    - The number 1 should have the same height but \( \frac{1}{4} \) of the area instead.
Result
## Result

- **Database Schema**

<table>
<thead>
<tr>
<th>Students</th>
<th>Reservations</th>
<th>Prices</th>
<th>Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>email</td>
<td>plug_id</td>
<td>date</td>
</tr>
<tr>
<td>dorm_name</td>
<td>start_date</td>
<td>price</td>
<td>energy</td>
</tr>
<tr>
<td>dorm_id</td>
<td>end_date</td>
<td></td>
<td>email</td>
</tr>
<tr>
<td>balance</td>
<td>qrcode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>plug_id</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>completed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time_remaining</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Result

#### API Endpoints

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/users/{email}/dorm-info (GET)</td>
<td>to get the user’s dorm information</td>
</tr>
<tr>
<td>/users/{email}/balance (GET)</td>
<td>to get the user’s balance</td>
</tr>
<tr>
<td>/users/{email}/transactions (GET)</td>
<td>to get the user’s payment history</td>
</tr>
<tr>
<td>/create-checkout-session (POST)</td>
<td>to create a checkout session for the user to top up money</td>
</tr>
<tr>
<td>/payment (POST)</td>
<td>to record the payment by the user after checking out and update the users balance</td>
</tr>
<tr>
<td>/washers/{washer_id}/price (GET)</td>
<td>to get the price of individual washers</td>
</tr>
<tr>
<td>/email (POST)</td>
<td>to send an email notification upon completion of laundry</td>
</tr>
<tr>
<td>/energy (PUT)</td>
<td>to update the energy usage of a person for a specific day</td>
</tr>
</tbody>
</table>
## Result

- **API Endpoints**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/reservation/check (GET)</td>
<td>to check if the given dates given by the user overlaps with existing reservations</td>
</tr>
<tr>
<td>/reservation (POST)</td>
<td>to insert a new reservation</td>
</tr>
<tr>
<td>/reservations/{email} (GET)</td>
<td>to retrieve the list of reservations made by the user (to show in the app)</td>
</tr>
<tr>
<td>/reservation/{qr_code} (DELETE)</td>
<td>to delete a reservation</td>
</tr>
<tr>
<td>/reservation/{qr_code}/time (GET)</td>
<td>to get the current time remaining of the washer for that reservation</td>
</tr>
<tr>
<td>/reservation/{qr_code}/time (PUT)</td>
<td>to update the time remaining of the washer for that reservation</td>
</tr>
<tr>
<td>/reservation/{qr_code}/status (PUT)</td>
<td>to update the reservation status e.g. reservation changes to in_progress after QR code validation.</td>
</tr>
</tbody>
</table>
Result

• Raspberry Pi Script
  • search_qrcode.py
    • Runs in an infinite loop to search for QR codes using the Python pi camera and pyzbar library
    • Upon scanning a QR code, sends a API call to validate it
    • Returns a response indicating the validity.
      • If valid, the script will hand the control to start_ocr.py to start the time extraction process and get a response of the duration of the reservation
      • Otherwise continue scanning
  • start_ocr.py
    • Similar but runs only for the reservation period
    • Contains the image pre-processing
    • Runs the SSOCR using the processed image and sends the time to the server then to the user application
```python
from PIL import Image
from pyzbar.pyzbar import decode
import io
import requests
import os
import json
from tuyac_connector import TuyaConnector

api_key = os.environ.get('TUYA_API_KEY')
api_secret = os.environ.get('TUYA_API_SECRET')
base_url = os.environ.get('BASE_URL')
connector = TuyaConnector(api_key, api_secret)

with picamera.PiCamera() as camera:
    # Continuously scanning for QR codes
    while True:
        camera.start_preview(fullscreen=False, window=(0, 0, 320, 240))
        camera.resolution = (1824, 768)
        stream = io.BytesIO()
        camera.capture(stream, format='jpeg')
        stream.seek(0)
        image = stream.read()
        image = Image.open(io.BytesIO(image))
        # Decode QR code
        for qr_code in decode(image):
            qr_code = qr_code.data.decode('utf-8')
            url = f'{base_url}/reservation/{qr_code}/status'
            data = {'status': 'in progress'}
            response = requests.put(url, json=data, headers={
                'Content-Type': 'application/json'})
            # If 204 i.e. correct QR code then students can start using washers
            if response.status_code == 204:
                # Switch on smart plug
                command = {'code': 'switch_1', 'value': True}
                response = connector.send_command(response.device_id, command)
                if response['success']:
                    # Start extracting time using the second script
                    # Similar to this with the image preprocessing steps
```
Result

- Sign In page
Result

• Each page has 3 components: a top bar, side and main content area
Result

- The header contains the user’s email, dorm name, their balance and a sign out button
- The information in the header is retrieved from the database
Result

- The default page loaded is the “Washers” page i.e., it is the home page
- Contains the list of washers located in a dorm
- Each washer section contains the plug id, the price and the availability status
- If a washer is currently reserved, the text is highlighted in red
Result

- Upon clicking one of the washers, a reservation form is brought out.
- Users can select the date and time of their choice
- An API is called to the server to check if it’s a valid reservation
Result

- A valid reservation must have:
  - No fields empty
  - Start time precedes the end time
  - Start time must be later than the current time
  - Not overlap with other reservations
- The text below will turn green if its valid
The reservation lists show the reservation history of a user. Each contains the plug ID, date of reservation, and status. Each reservation can be “Incomplete”, “In Progress”, “Completed.”
Result

- Clicking on a “Incomplete” reservation will show a QR code
- The QR code value is a 128-bit key to identify unique reservations
Result

- Clicking on a “In Progress” reservation shows the time remaining of the washer
- A “Complete” reservation does not show anything and can be deleted using the delete button
- It will be removed from the database
Result

- The "Add Payment" page allows users to top up their account using Stripe.
- Users can top up HKD20, HKD 50 or HKD 100.
- A payment history is shown below with each row having the Stripe payment id, amount and date of payment.
Result

• A checkout page is shown when users click on a price
• A checkout session is given from the backend to the frontend
• Upon completing payment, an API in the backend is called by Stripe to update the database
Result

- Energy Consumption page shows the daily energy usage of students
- Retrieved from the smart plug API
- Includes total and average energy use

Other Information
- Energy consumption has increased by 20% compared to last week.
- The highest energy consumption was recorded on 4/16.
- Energy consumption during the day is higher than during the night.
Result

• Email notification is sent when the Pi Camera detects 0:00
• Pi sends an API call to the backend
• Sends via Python smtp library
Conclusion

• HKU dorm residents are busy enough

• Inconvenient laundry does not have to add to their worries

• Created an inexpensive solution to reduce overhead

• Future plans include making a mobile version and possibly integrating other devices such as AC and printers to the app