Team 26: Zipcart

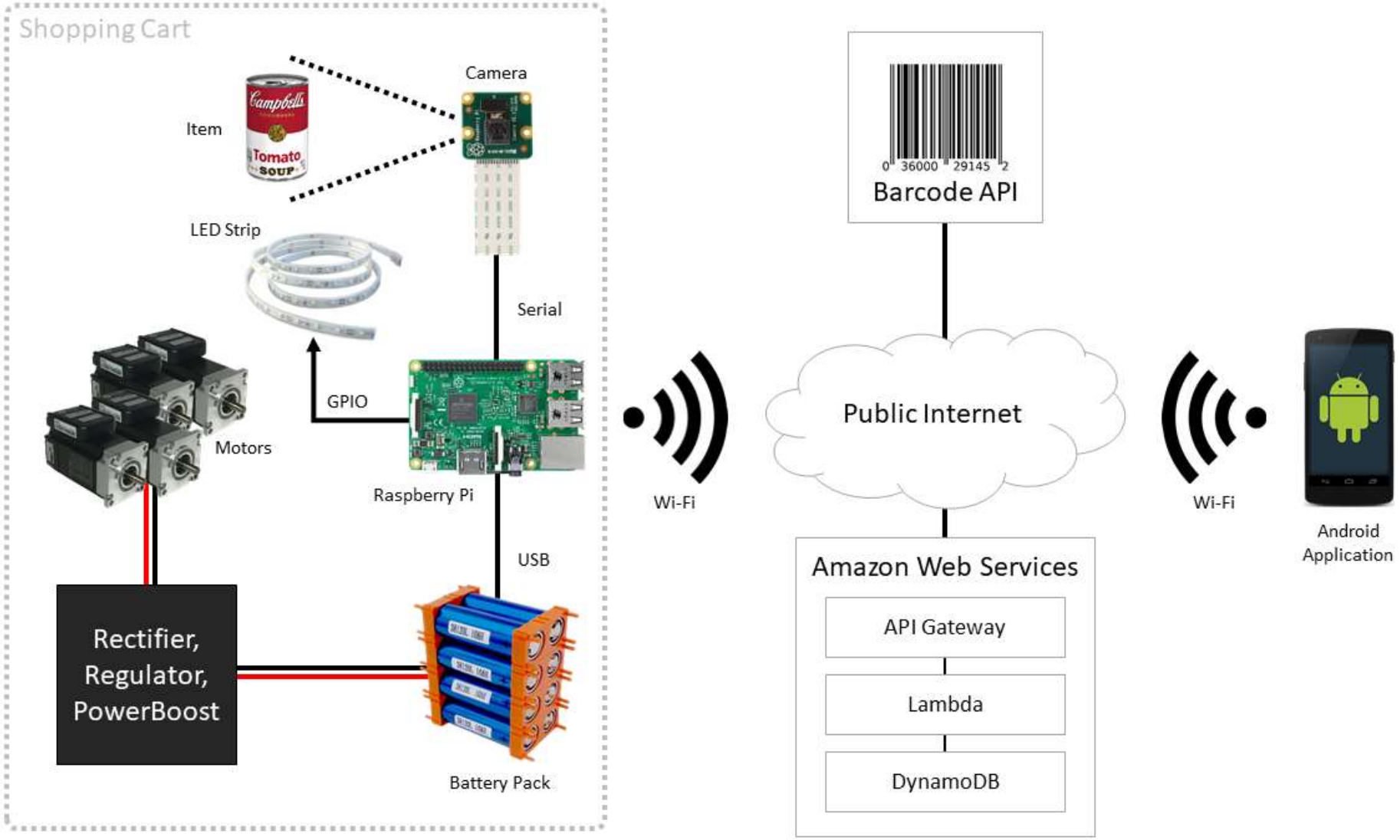
Comprehensive Design Review

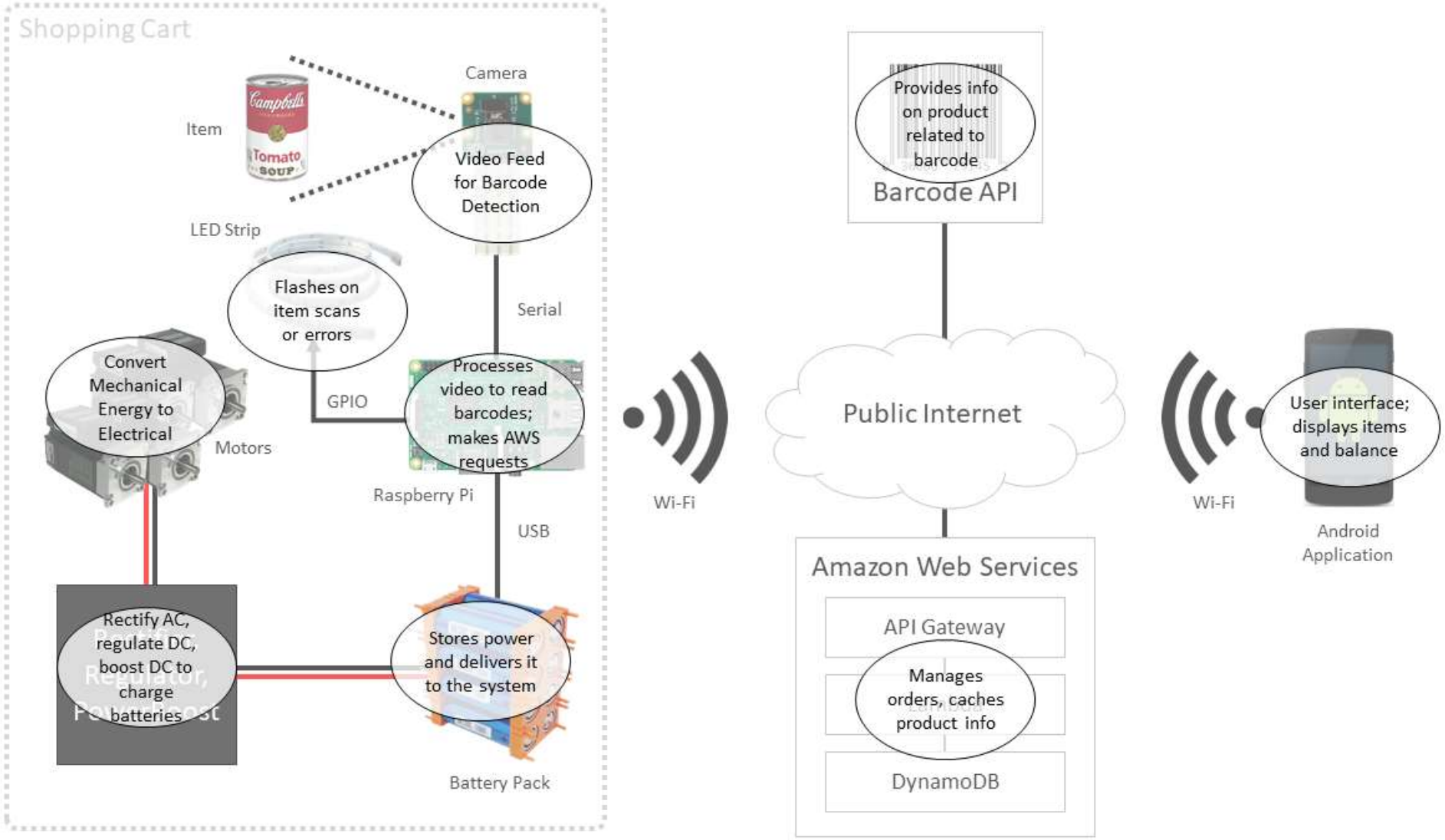
Team

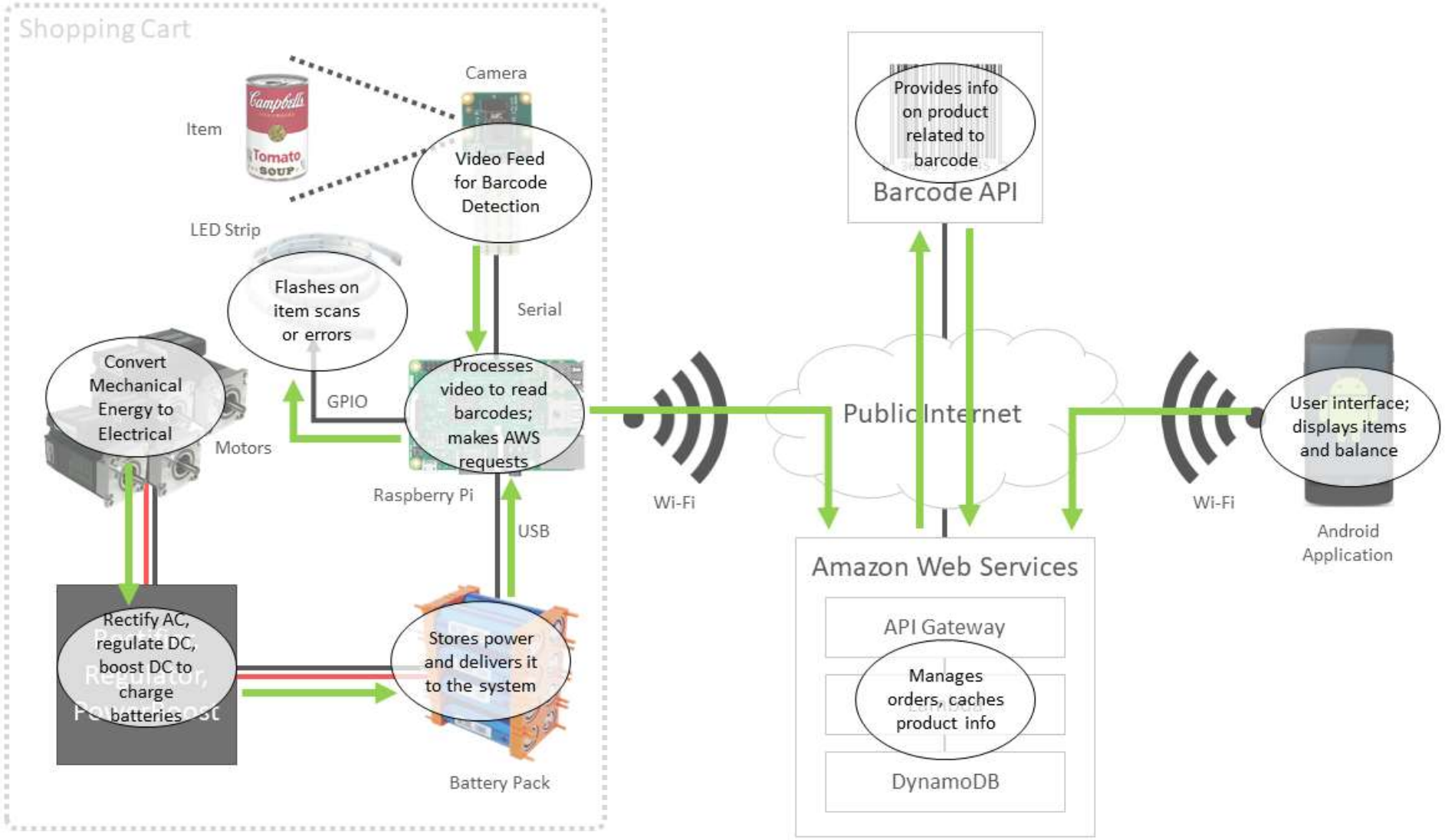
Ryan Lagasse

Ricardo Henriquez

Jonathan Azevedo


Shopping Cart
Item
Camera
LED Strip
Serial
GPIO
Motors
Raspberry Pi
USB
Rectifier, Regulator, PowerBoost
Battery Pack
Wi-Fi
Barcode API
Public Internet
Amazon Web Services
API Gateway
Lambda
DynamoDB
Wi-Fi
Android Application


Shopping Cart
Item
Camera
Video Feed for Barcode Detection
LED Strip
Flashes on item scans or errors
Serial
Convert Mechanical Energy to Electrical
Motors
GPIO
Processes video to read barcodes; makes AWS requests
Raspberry Pi
USB
Rectify AC, regulate DC, boost DC to charge batteries
Stores power and delivers it to the system
Battery Pack
Wi-Fi
Provides info on product related to barcode
Barcode API
Public Internet
Amazon Web Services
API Gateway
Manages orders, caches product info
DynamoDB
Wi-Fi
User interface; displays items and balance
Android Application

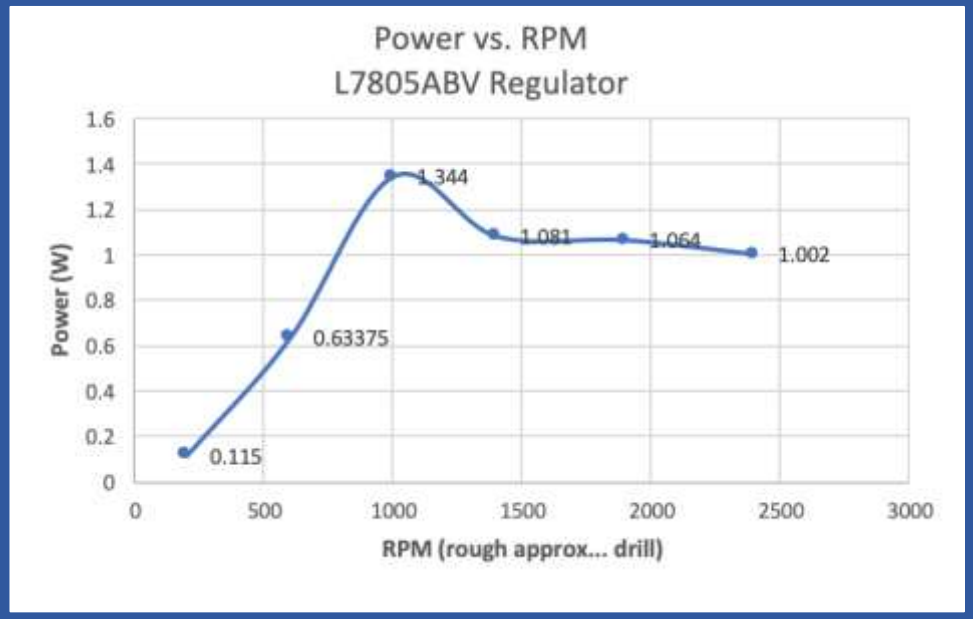

Shopping Cart
Item
Camera
Video Feed for Barcode Detection
LED Strip
Flashes on item scans or errors
Serial
Convert Mechanical Energy to Electrical
Motors
GPIO
Processes video to read barcodes; makes AWS requests
Raspberry Pi
USB
Rectify AC, regulate DC, boost DC to charge batteries
Stores power and delivers it to the system
Battery Pack
Provides info on product related to barcode
Barcode API
Wi-Fi
Public Internet
Wi-Fi
User interface; displays items and balance
Android Application
Amazon Web Services
API Gateway
Manages orders, caches product info
DynamoDB

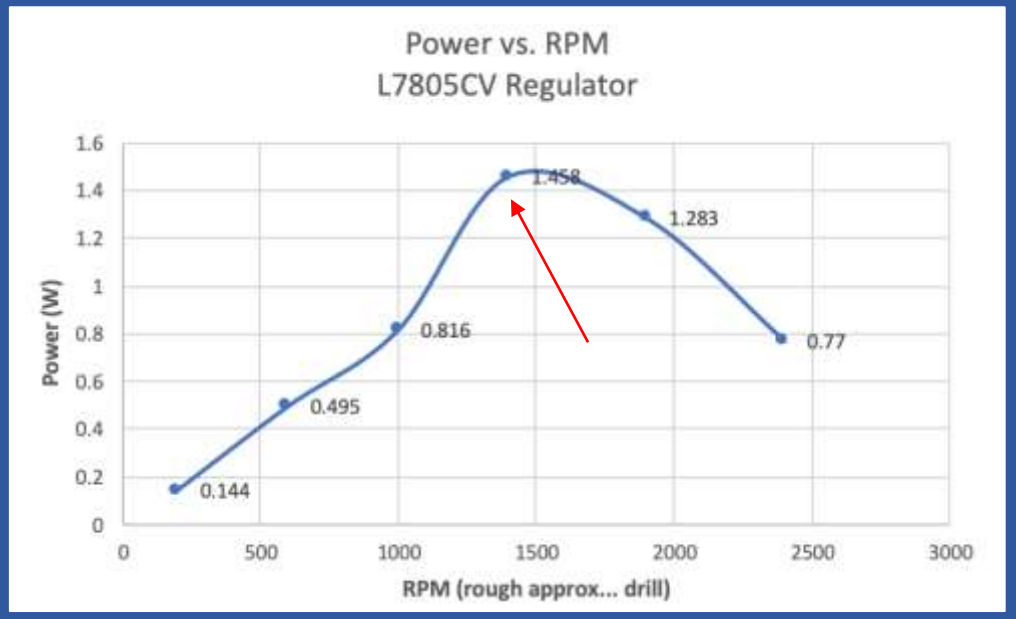
CDR Deliverables

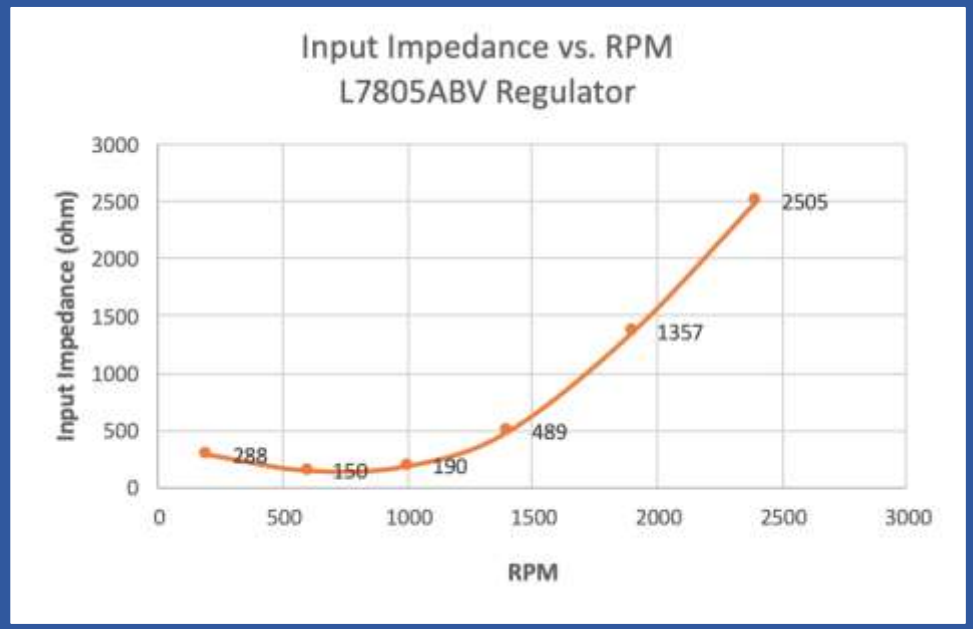
- ✓ Mount system on a shopping cart
- ❑ Detect barcodes fully around products
- ❑ Remove items as they exit the cart
- ✓ Increase power delivered to system
- ✓ Create PCB for the power circuit
- ✓ Make a fully-featured interface

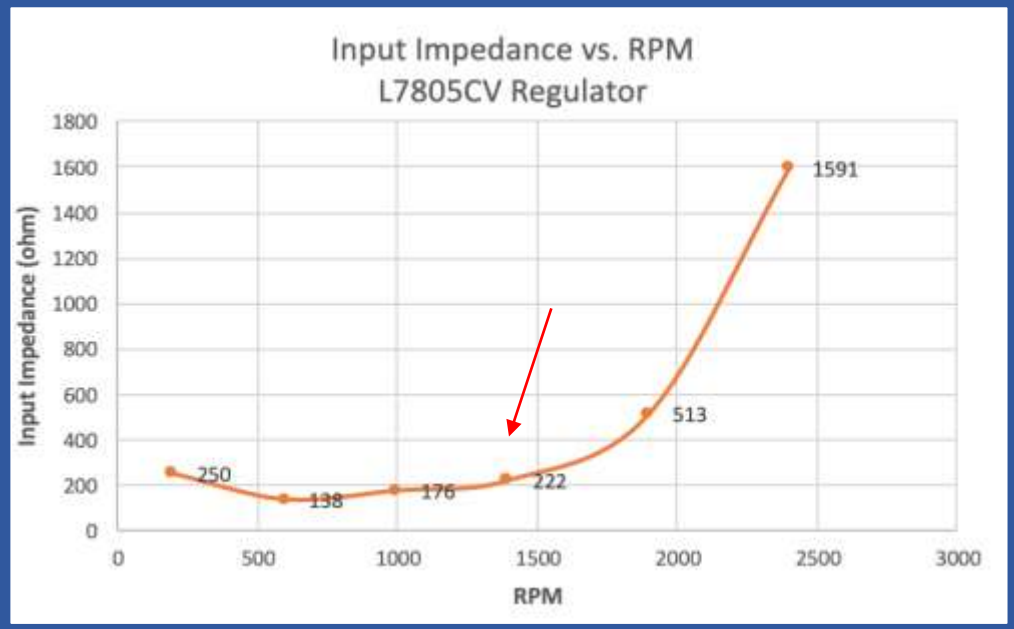
Motor Experiments

- Goal was determine the peak power performance of a single motor
- **Procedure**
  - Used a drill to rotate motor shaft of a single motor at varying speeds
  - Motor was loaded by entire circuit + tested different regulators
  - Measured current and voltage of regulator to calculate power
- **Results**
  - L7805ABV = 1.344W @ 1200-1600RPM (approximately)
  - L7805CV = 1.458W @ 1200-1600RPM (approximately)**
- **Pi Consumption**
  - Standby → 2.5W, 5V, 500mAh
  - All Peripherals → 4W, 5V, 800mAh


Power vs. RPM
L7805ABV Regulator
Power (W)
RPM (rough approx... drill)
0.115
0.63375
1.344
1.081
1.064
1.002


Power vs. RPM
L7805CV Regulator
Power (W)
RPM (rough approx... drill)
0.144
0.495
0.816
1.458
1.283
0.77


Input Impedance vs. RPM
L7805ABV Regulator
Input Impedance (ohm)
RPM
288
150
190
489
1357
2505

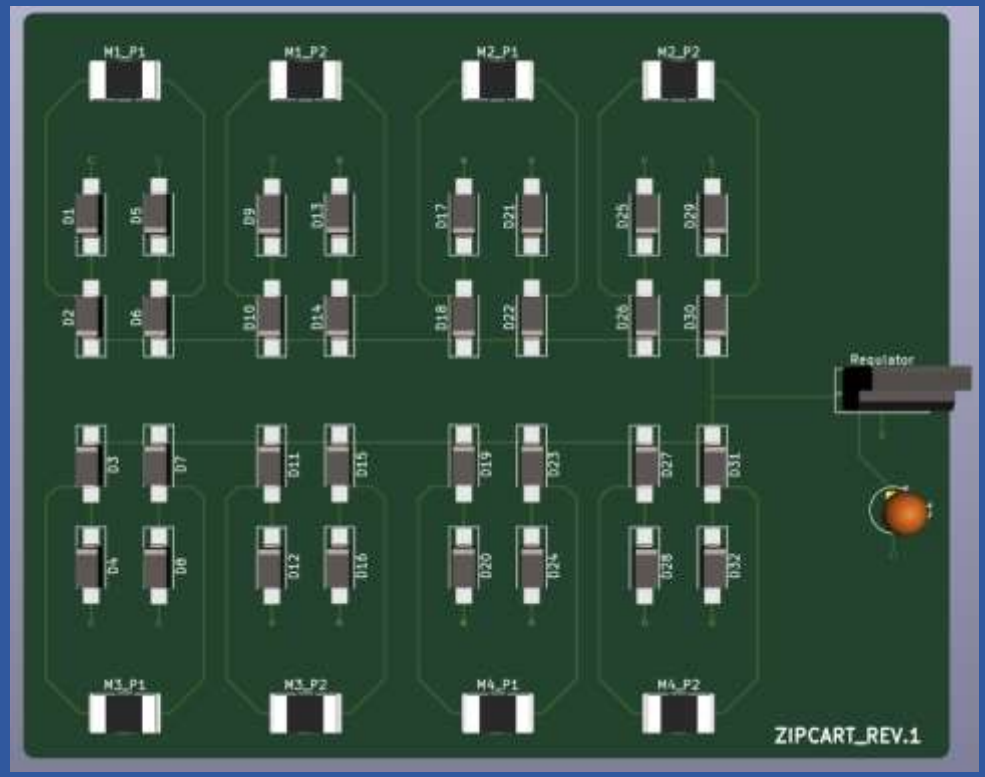

Input Impedance vs. RPM
L7805CV Regulator
Input Impedance (ohm)
RPM
250
138
176
222
513
1591

Gears

- Peak power performance @ 1400 – 1600 RPM
- Average walking speed: 200 RPM
- Designed an 8:1 gear ratio to achieve maximum performance
  - $Teeth_A$ = 64, $Teeth_B = 8$
  - $\frac{Teeth_A}{Teeth_B} = \frac{RPM_A}{RPM_B} = 8$ Gear Ratio
- Gear designed using AutoCAD & 3D printed in M5

PCB

- Dimensions: 4.10 x 3.30 inches
- Wires four motors in parallel to increase power produced
- Currently being fabricated; expected delivery on April 1st



System Software Overview

- Detection Module (C++) <detect.cpp>

  Process frames of video stream to read item barcodes

- AWS Request Handler (Python) <request.py>

  Interact with the AWS order database through API requests

- Feedback Controller (Python) <feedback.py>

  Signifies system states to the shopper through LEDs

Feedback Controller States

- Steady Yellow

  System is waiting for QR code to **synchronize with user interface on order ID**

- Flashing Green

  System has read the barcode of an **item to be added**

- Flashing Orange

  System has read the barcode of an **item to be removed**

- Flashing Red

  System has detected that an **item was not successfully processed**

Detection Issues

Accuracy

Range, dependability of scan success

- Best to post-process stream on laptop
- Fairly accurate on Python
- Unquantified success: single-threaded C++
- Zero success yet: multi-threaded C++

Performance

Frame processing throughput

- Slow on Python (no parallel processing)
- Fast: single-threaded C++
- Faster: multi-threaded C++
- Fastest: single-threaded C++ on laptop

Trials with Laptop Post-Processing

Procedure

1. Take **raw footage** of desired resolution on Raspberry Pi
2. Copy footage over to laptop, convert to MP4
3. Process footage through ZBar, write detection boxes to video

Results

Observed detection between **fourteen and twenty-two inches**, still.
Up to **twenty inches** while slowly placing items into cart.

C++ Implementation

Issue

Python applications cannot be parallelized (only one core / time)

Assessment

Due to performance metrics and system resource constraints, we need to parallelize frame processing.

Decision

Re-implement detection module in C++

Single-Threaded Performance Comparison

Test

In one thread, grab one thread then process it iteratively.

Use same OpenCV API functions in both applications.

Run on Raspberry Pi.

Results

Python: 1.45 FPS

C++: 1.85 FPS

Task-Decoupled Performance Comparison

Test

In a single thread, grab N = 300 frames and insert them into a queue.

Then, process the N frames until the queue is depleted.

Compare Raspberry Pi to a more performant system (laptop, no GPU).

Results	Raspberry Pi	Dell Inspiron i5 Laptop
Producer	5.35 FPS / 56 seconds	15.51 FPS / 19 seconds
Consumer	3.15 FPS / 95 seconds	113.58 FPS / 2.6 seconds

Detection Approach for FPR

- Consider purchasing a more powerful computing platform
- Work on issues with multi-threaded accuracy

  Need to perform more debugging to find root cause

Interface Specifications

- Start a new order
- View balance and list of items in the order in near-real time
- Process payment
- Complete transaction

Graphical User Interface

Demo Overview

- Servo mounting, gears
- Emulated system demo with functional user interface (no detection)
- Experimental detection samples and measurements
- Versions of detection implementation, tradeoffs, and approach
- Q&A

FPR Deliverables

1. Fix detection
2. Remove items as they exit the cart
3. Populate PCB
4. Wire motors, battery, and Pi to cart
5. Integrate product info into app