UMassAmherst
The Commonwealth's Flagship Campus

Team 13 - SmartRack

Midway Design Review

December 5th, 2019

Team Roles

Arthur, CSE, Mobile Application Development

Alessy, EE, Manager, RFID & Raspberry Pi Development

Fedor, EE, PCB Lead, PCB Development

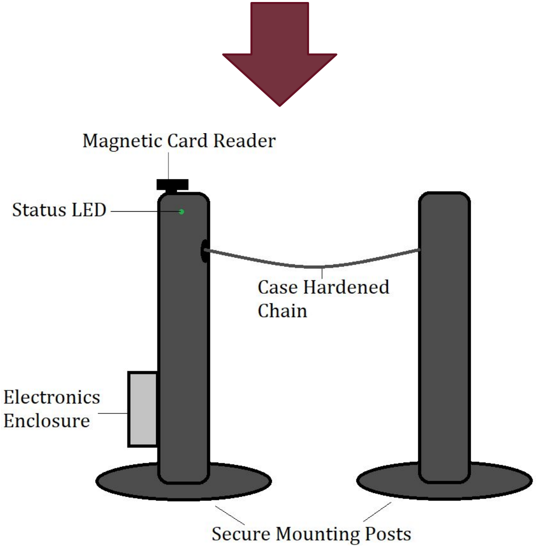
Andrew, EE, Hardware Fabrication

Problem Statement

- Bike racks are typically full and unorganized
- No guarantee of a spot

Problem Statement: Solution

- SmartRack!
- Reserve bike rack ahead of time
- Real-time feedback on bike rack availability
- Lock and unlock with UCard



Current Alternatives

Bikeep

Pros:

- Solar Powered (Optional)
- Secure Bar Locking

Cons:

- Expensive (> $1000)
- No reservations
- Must buy RFID card to use without phone

vadeBike

Pros:

- Chain Lock
- Small Storage Space

Cons:

- No Mobile App
  - No Reservations
- Pay for Spot
- Geographical Limitation

Bikeep

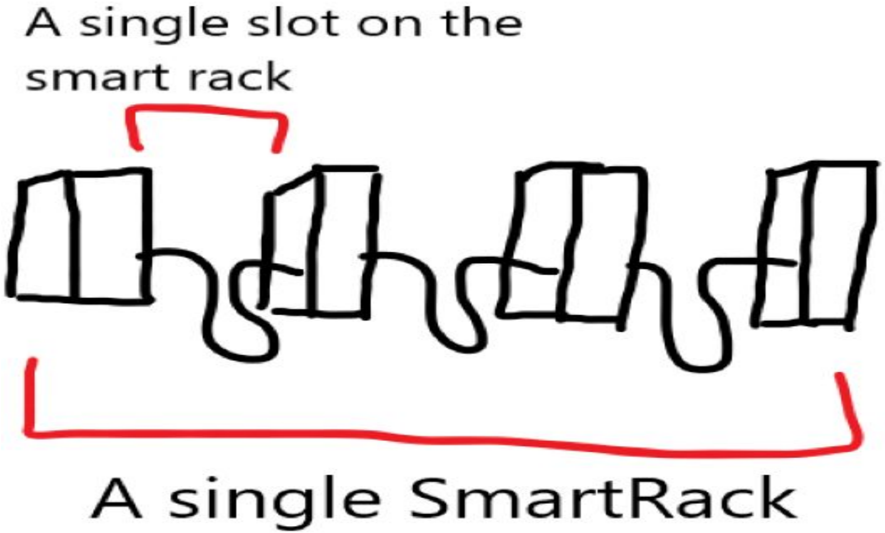
vadeBike

Current Alternatives Comparison

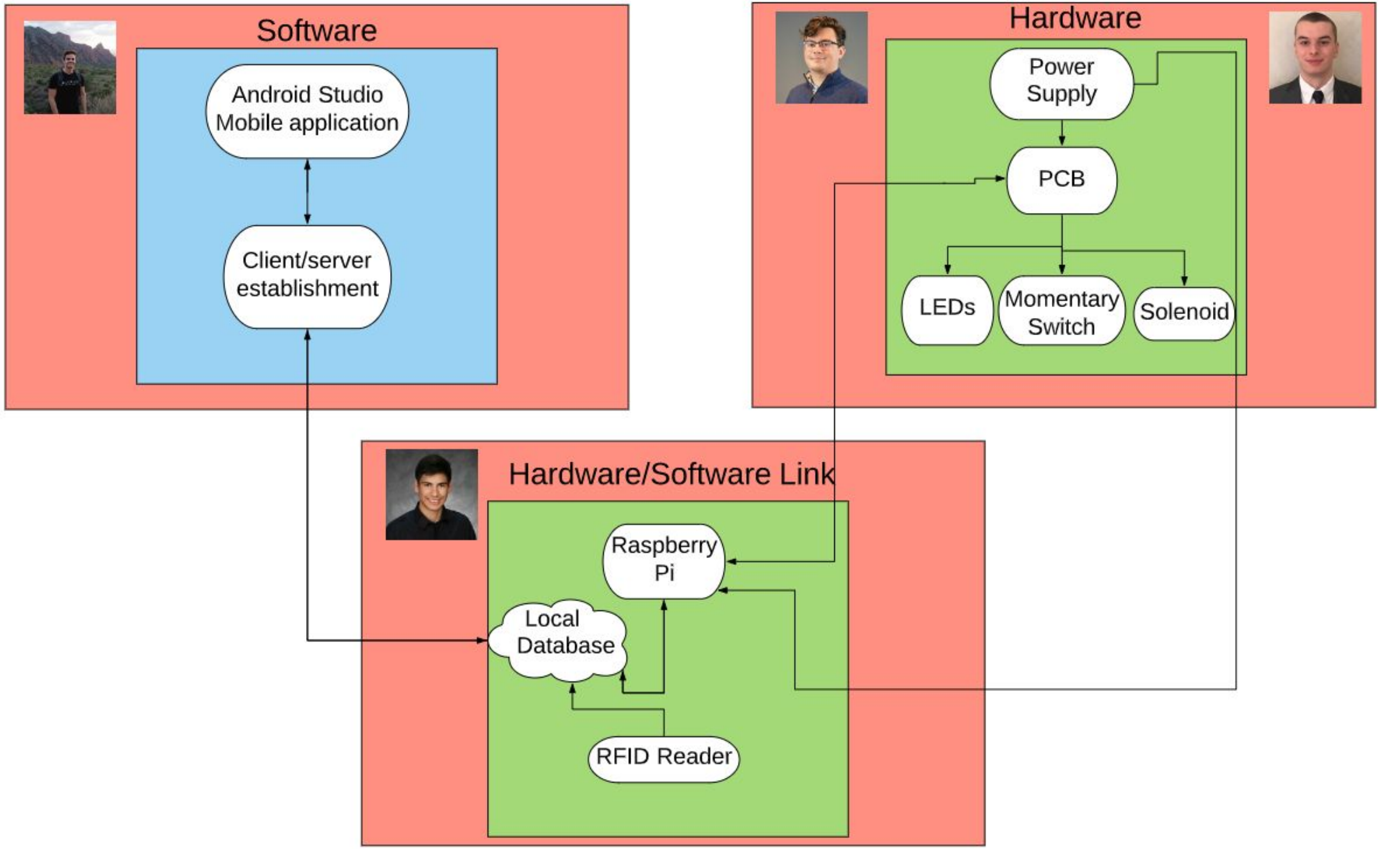
	Bikeep	vadeBike	SmartRack
Mobile App	YES	NO	YES
Reservations	NO	NO	YES
Free for User	NO	NO	YES
Power Source	Solar or AC Power (220V/24V)	AC Power (220V/24V)	AC Power (120V)

System Specifications

- Mobile application reservation
    - Exclusive to SMART Rack
- 15 minute grace period
- Embedded Locking System
- RFID Compatibility
- Raspberry Pi



Block Diagram



Software

Android Studio Mobile application

Client/server establishment

Hardware

Power Supply

PCB

LEDs

Momentary Switch

Solenoid

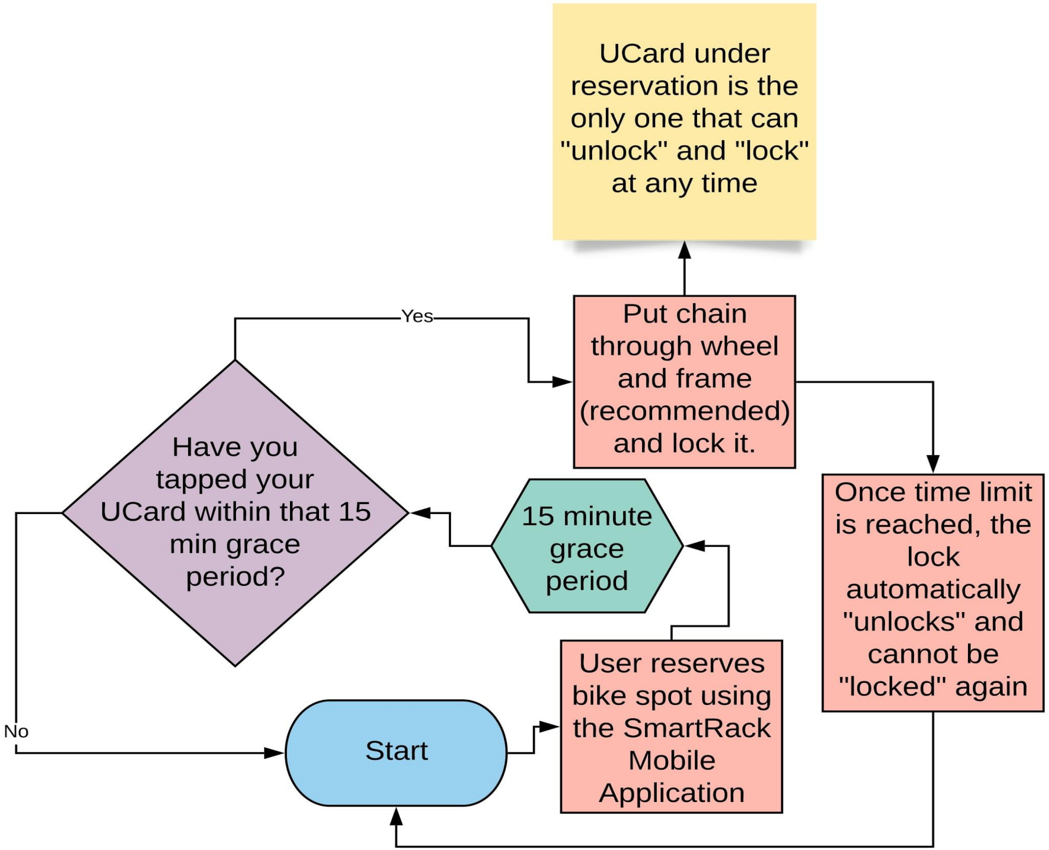
Hardware/Software Link

Raspberry Pi

Local Database

RFID Reader

State Machine Diagram



Start

User reserves bike spot using the SmartRack Mobile Application

15 minute grace period

Have you tapped your UCard within that 15 min grace period?

Put chain through wheel and frame (recommended) and lock it.

UCard under reservation is the only one that can "unlock" and "lock" at any time

Once time limit is reached, the lock automatically "unlocks" and cannot be "locked" again

MDR - Deliverables

- Communication link between the Raspberry Pi and the Mobile Application

  Complete

- Configuration tool setup
  - MagStripe Reader data can be manipulated

  Complete, changed to RFID Reader

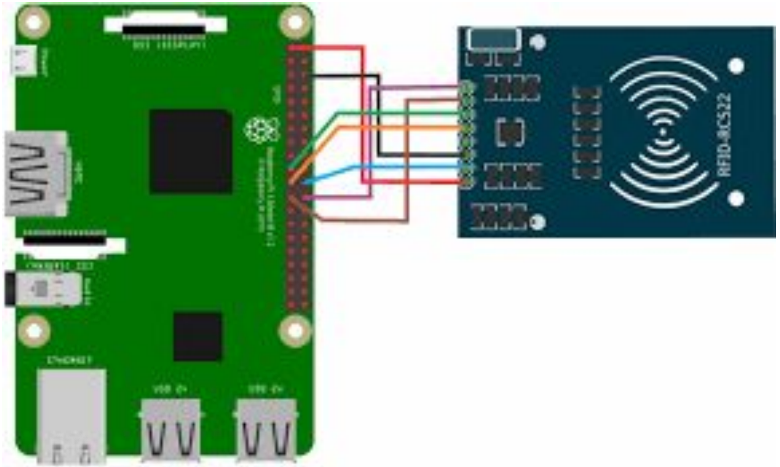
- Set up local database to support all incoming data

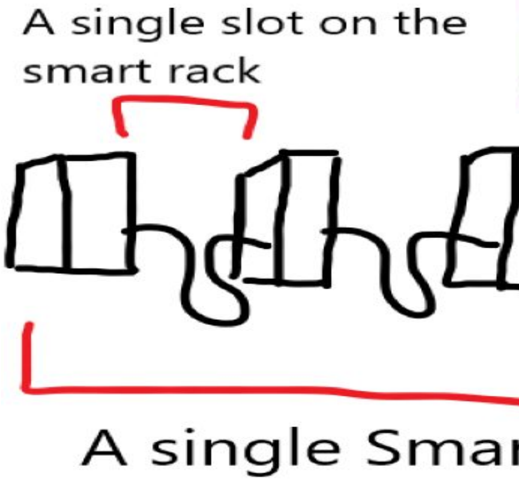
  Complete

RFID Compatibility

- Configuration tool (RFID) is setup and ready to manipulate data.

- Currently works for a single slot of the SmartRack

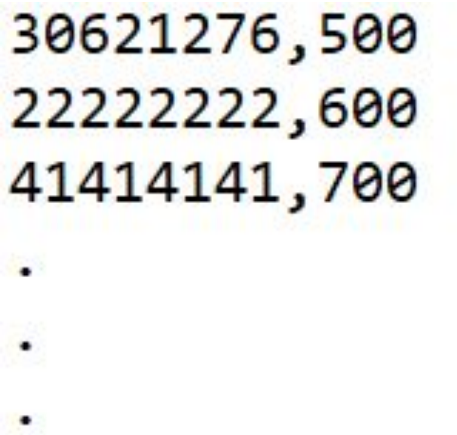




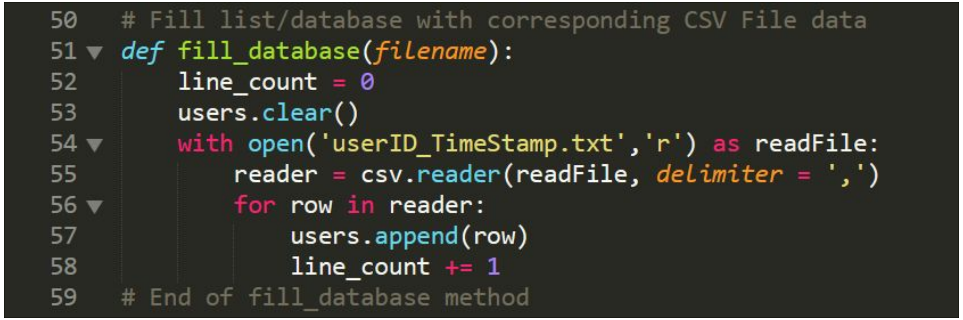
Raspberry Pi Local Database

- Local database supports incoming user data

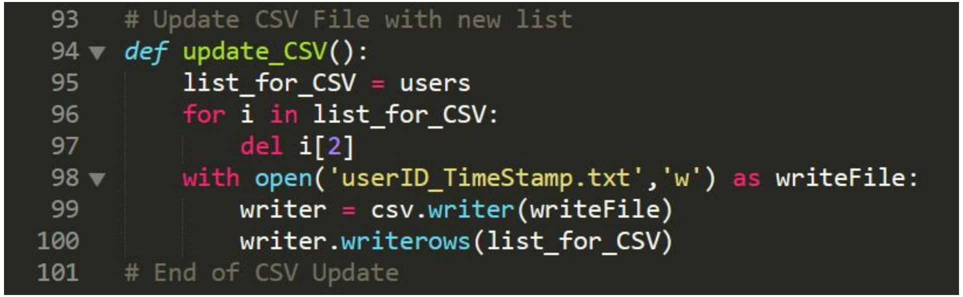
- CSV File -> Python List -> CSV File



```
30621276,500
22222222,600
41414141,700
.
.
.
```



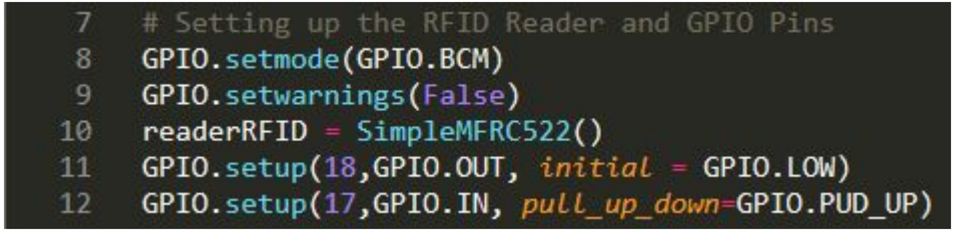
```
# Fill list/database with corresponding CSV File data
def fill_database(filename):
    line_count = 0
    users.clear()
    with open('userID_TimeStamp.txt','r') as readFile:
        reader = csv.reader(readFile, delimiter = ',')
        for row in reader:
            users.append(row)
            line_count += 1
# End of fill_database method
```



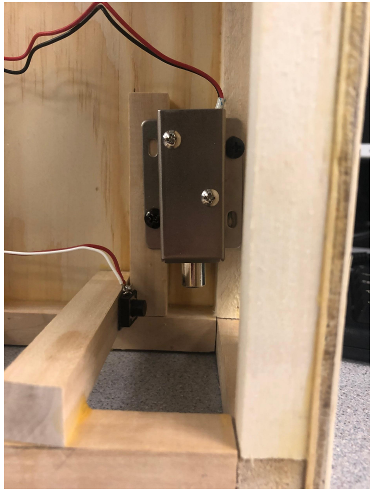
```
# Update CSV File with new list
def update_CSV():
    list_for_CSV = users
    for i in list_for_CSV:
        del i[2]
    with open('userID_TimeStamp.txt','w') as writeFile:
        writer = csv.writer(writeFile)
        writer.writerows(list_for_CSV)
# End of CSV Update
```

Raspberry Pi communication link with Breadboard

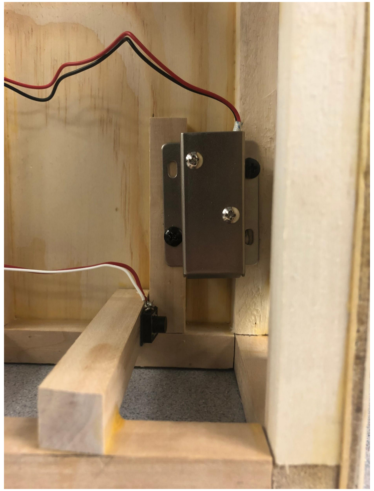
- GPIO Pin #17 goes *HIGH* only when RFID is accepted
  - 3.3V = *HIGH*



```
# Setting up the RFID Reader and GPIO Pins
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
readerRFID = SimpleMFRC522()
GPIO.setup(18,GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(17,GPIO.IN, pull_up_down=GPIO.PUD_UP)
```



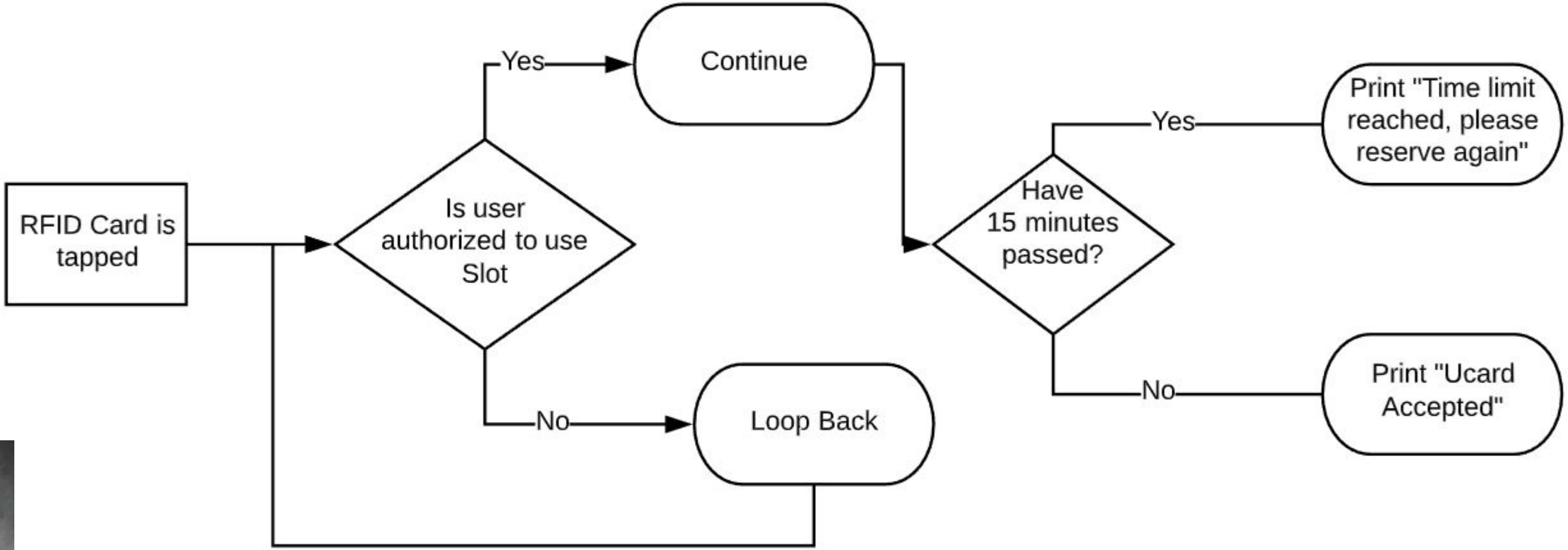
GPIO.LOW



GPIO.HIGH

Authorized vs Unauthorized User

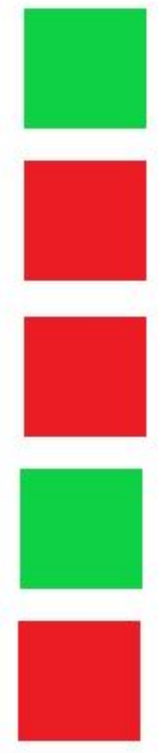
- If not an authorized user or a non-existent user, the GPIO pin remains *LOW*
  - Initial = GPIO.LOW



CDR: Multiple Slot Integration

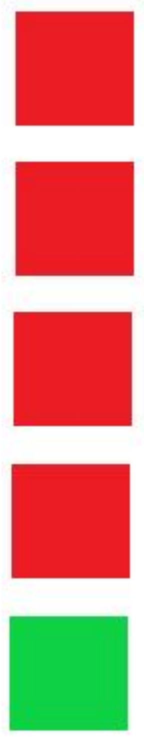
Plan A

- User will be able to choose which slot out of two slots from the SmartRack they want through the mobile application.
- Program should be able to differentiate which slot is authorized to be used for which user.



Plan B

- User will be automatically reserved for the next available slot in the rack (cannot pick specific slot)
- User will only be told which slot is available to them (less information for user to worry about)

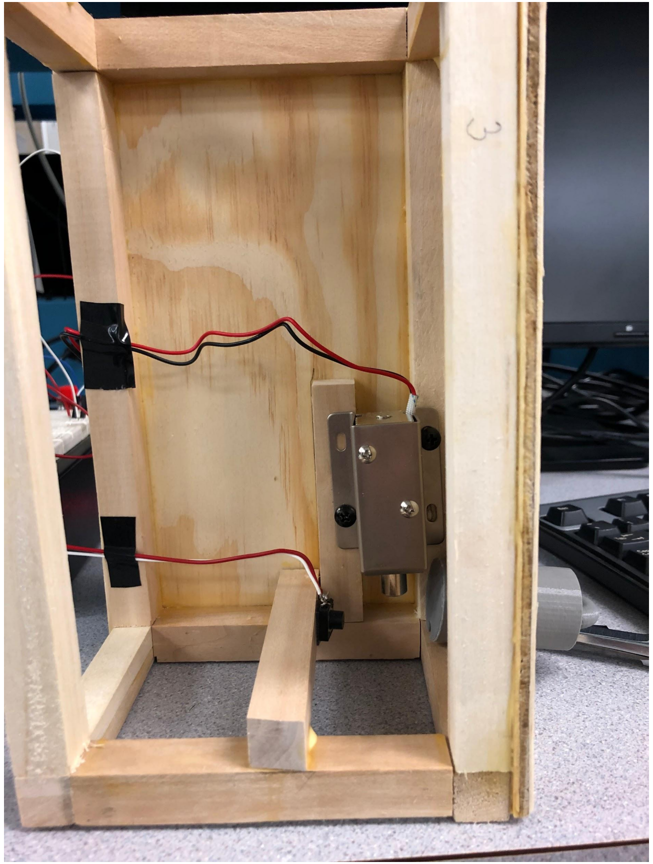


MDR Deliverables - Andrew

- Prototype model of the rack with custom components 3D printed Complete
- Lock mechanism operating time < 3 sec (Based on 50:1 gearing servo speed) Changed to Solenoid, Complete

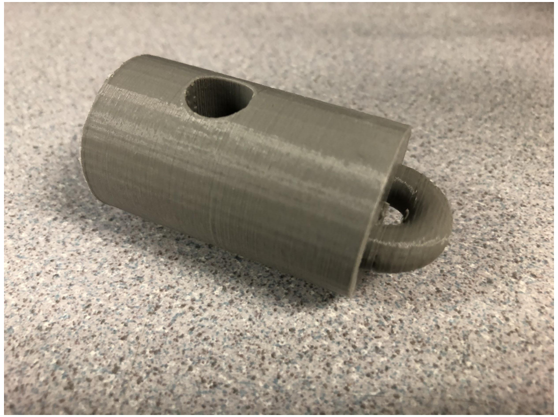
Prototype Frame Design

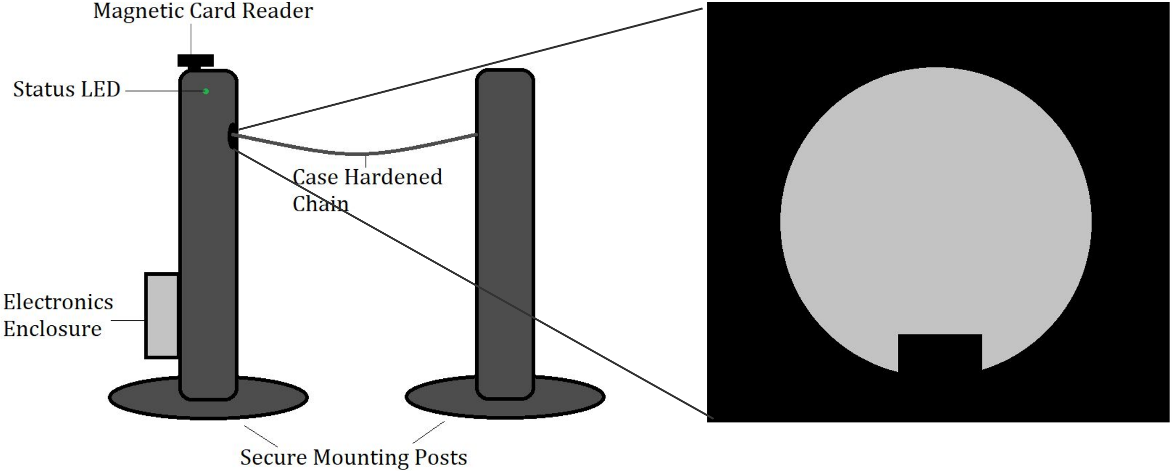
- Model serves as a proof of concept for final frame design
- Demonstrates how final frame will operate in conjunction with Raspberry Pi



Lock Design

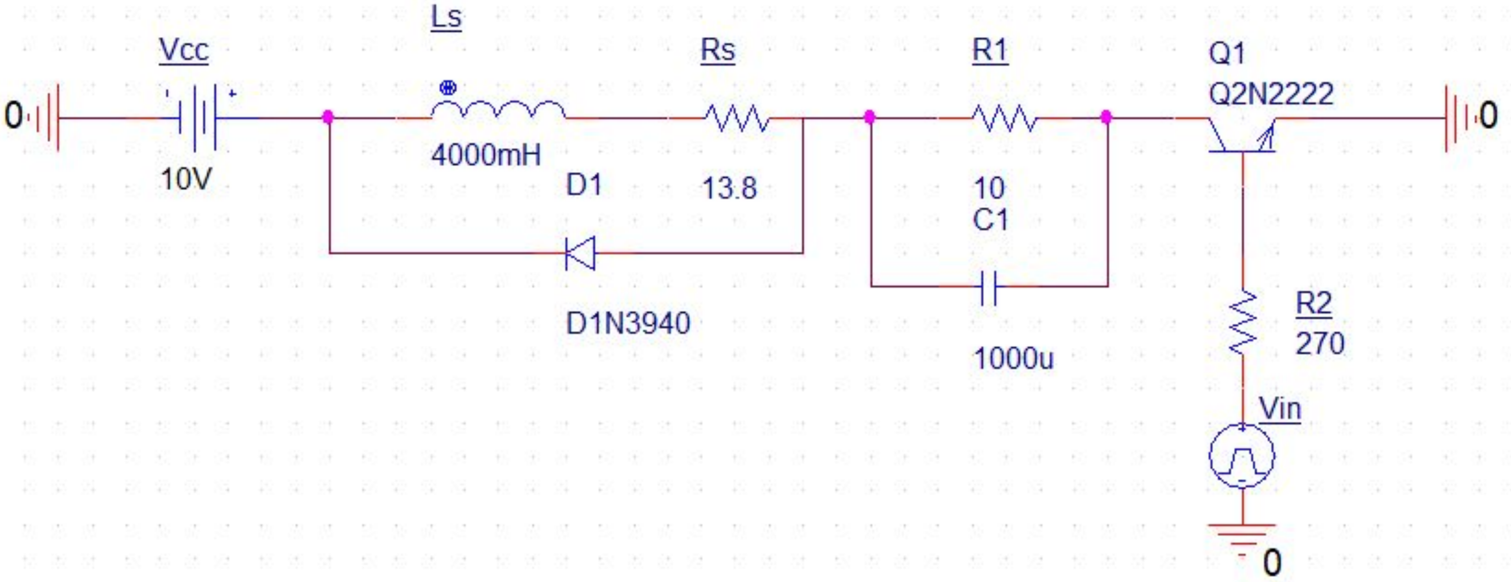
- Custom fabricated steel cylinder at the end of the chain
- User-friendly - grooved channel to make locking simple
- Rack will not unlock unless card is tapped





Solenoid Operation

- Solenoid takes relatively high voltage (~10V) to pull in metal pin
- Circuit design optimizes power consumption



MDR Deliverables - Fedor

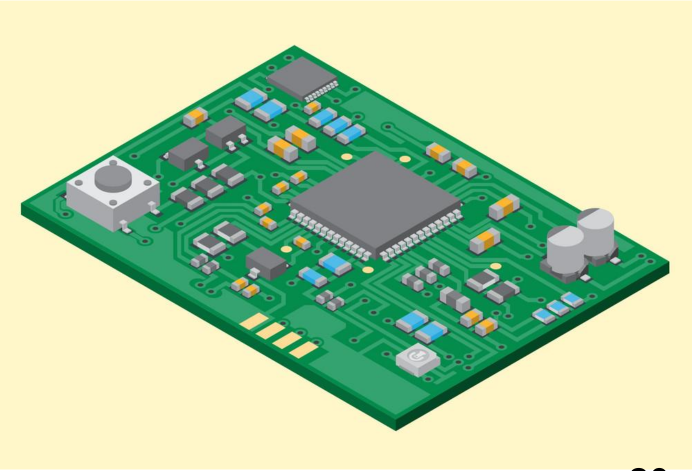
- PCB Prototype on a breadboad
  - Working interaction between components
    - LEDs Complete
    - Solenoid Complete
    - Switch Complete
  - Ability to regulate power between components

Partially complete- parts have not come in yet

PCBA Design

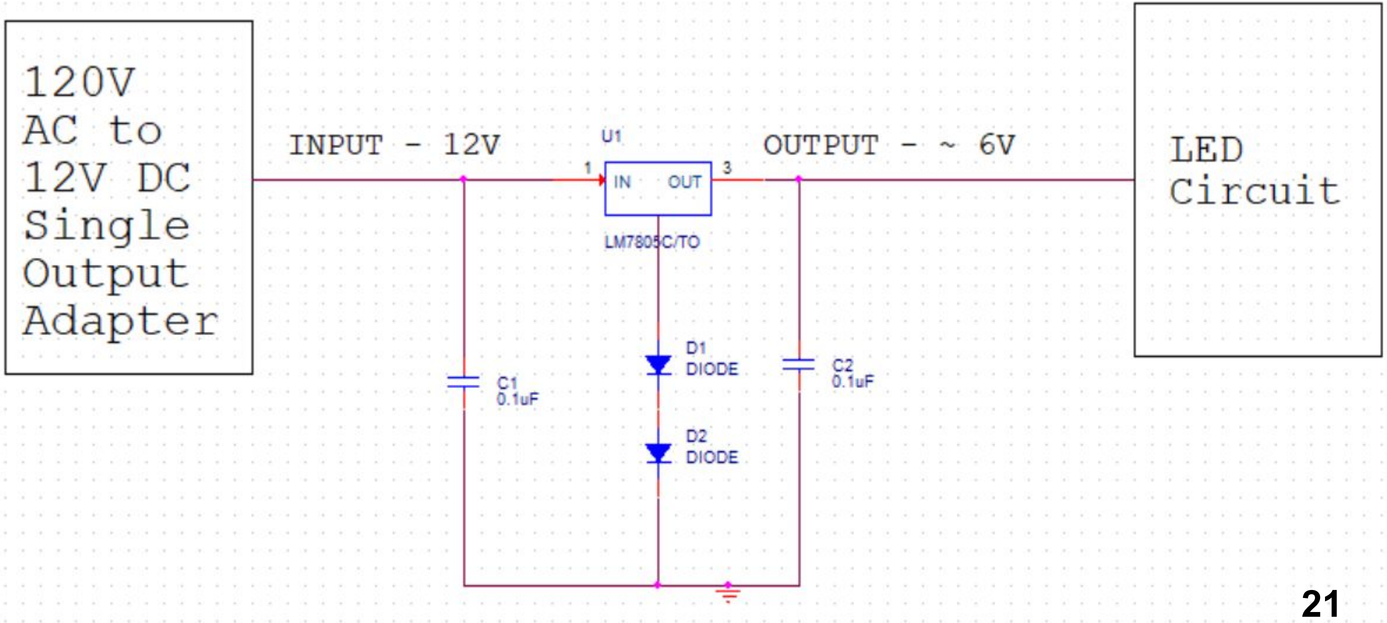
- Will be designed in Altium
- Interactions with all electrical components
  - LEDs
  - RFID Reader
  - Solenoid
- Controls Distribution of Power

ALTIUM DESIGNER



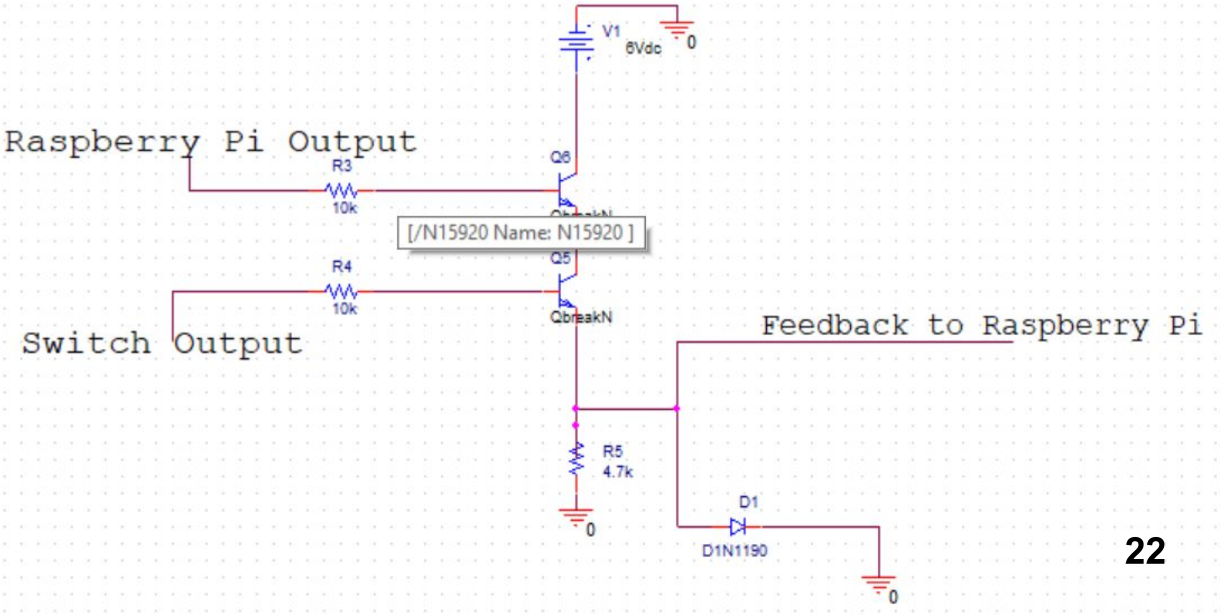
Power Regulation

- Low Power Consumption, <21W
- Power Supply - 12V 10A 120W
- Using LM7805 Voltage Regulator to step down voltage



Lock Status LED Circuit

- Utilizes communication between Lock and Raspberry Pi
- LED Status Indicator for User
  - Indicator if lock should be active
- Two transistors set up in series to make an AND gate
- When both transistors are in saturation, the output is HIGH, otherwise LOW
- LED status transmitted back to Pi



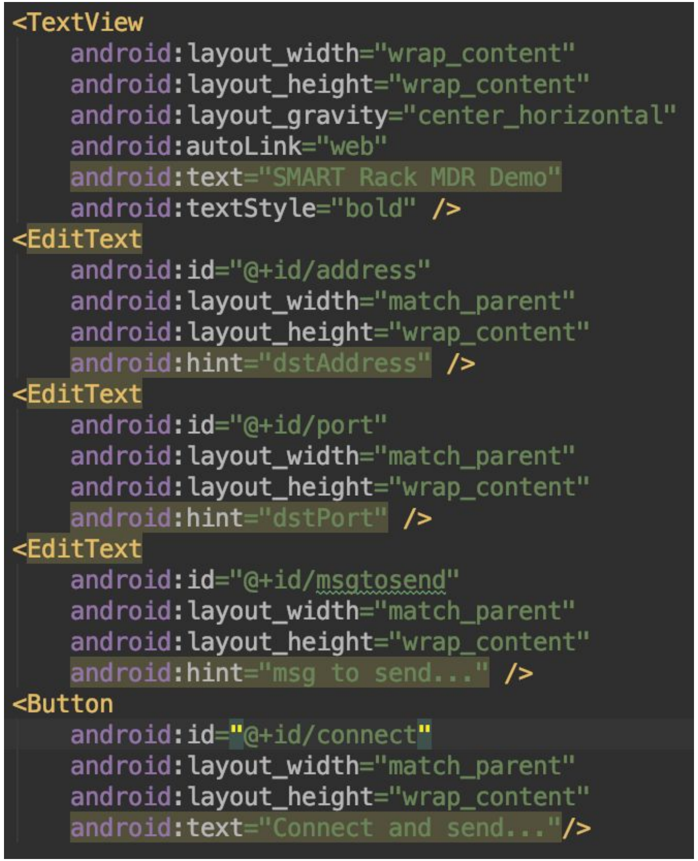
MDR Deliverables - Arthur

- Basic Android Studio Application that allows user to specify 8-digit ID to be transmitted Complete
- Server/Client backend that stores 8-digit student ID and timestamp in local database Complete

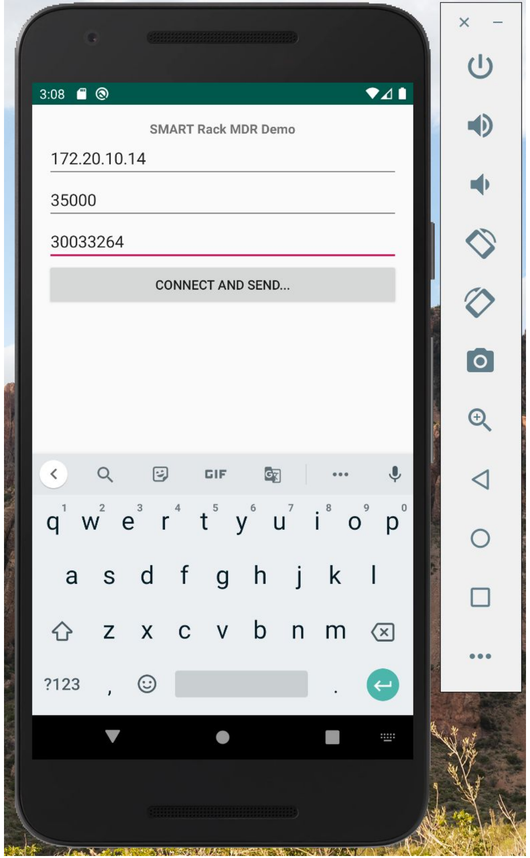
App Development

- Android Studio
  - UI - Linear Layout (next slide)
  - IP/Port specification
  - Prompts user to specify their student ID
  - One-way transmission of student ID/timestamp via server/client backend

App Development - Linear Layout



```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:autoLink="web"
    android:text="SMART Rack MDR Demo"
    android:textStyle="bold" />
<EditText
    android:id="@+id/address"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="dstAddress" />
<EditText
    android:id="@+id/port"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="dstPort" />
<EditText
    android:id="@+id/msgtosend"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="msg to send..." />
<Button
    android:id="@+id/connect"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Connect and send..."/>
```



3:08

SMART Rack MDR Demo

172.20.10.14

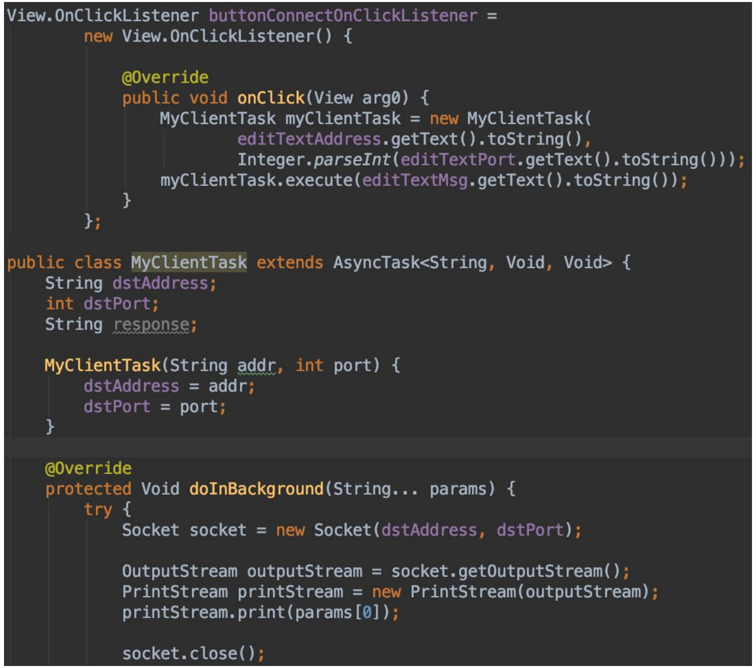
35000

30033264

CONNECT AND SEND...

Android App Networking - Client

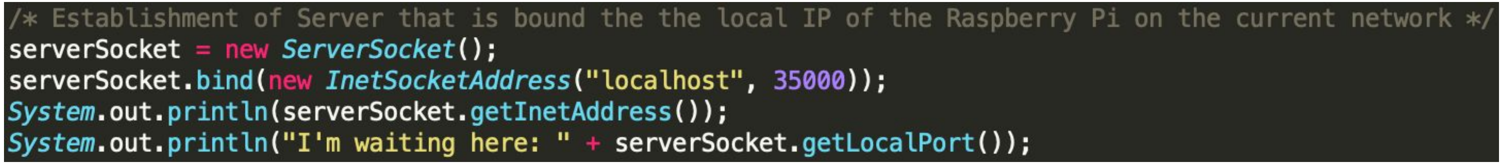
- Waits for user to click on "Connect and Send" button
  - On click, application sends data to server by executing "MyClientTask"



```
View.OnClickListener buttonConnectOnClickListener =
        new View.OnClickListener() {

            @Override
            public void onClick(View arg0) {
                MyClientTask myClientTask = new MyClientTask(
                        editTextAddress.getText().toString(),
                        Integer.parseInt(editTextPort.getText().toString()));
                myClientTask.execute(editTextMsg.getText().toString());
            }
        };

public class MyClientTask extends AsyncTask<String, Void, Void> {
    String dstAddress;
    int dstPort;
    String response;

    MyClientTask(String addr, int port) {
        dstAddress = addr;
        dstPort = port;
    }

    @Override
    protected Void doInBackground(String... params) {
        try {
            Socket socket = new Socket(dstAddress, dstPort);

            OutputStream outputStream = socket.getOutputStream();
            PrintStream printStream = new PrintStream(outputStream);
            printStream.print(params[0]);

            socket.close();
```

Raspberry Pi Networking - Server

- Server hosted on Raspberry Pi
  - Binds a socket to the localhost of Pi and specified port
  - Waits for connections from clients (hosted on Android App)
  - Stores student ID number and timestamp in local database



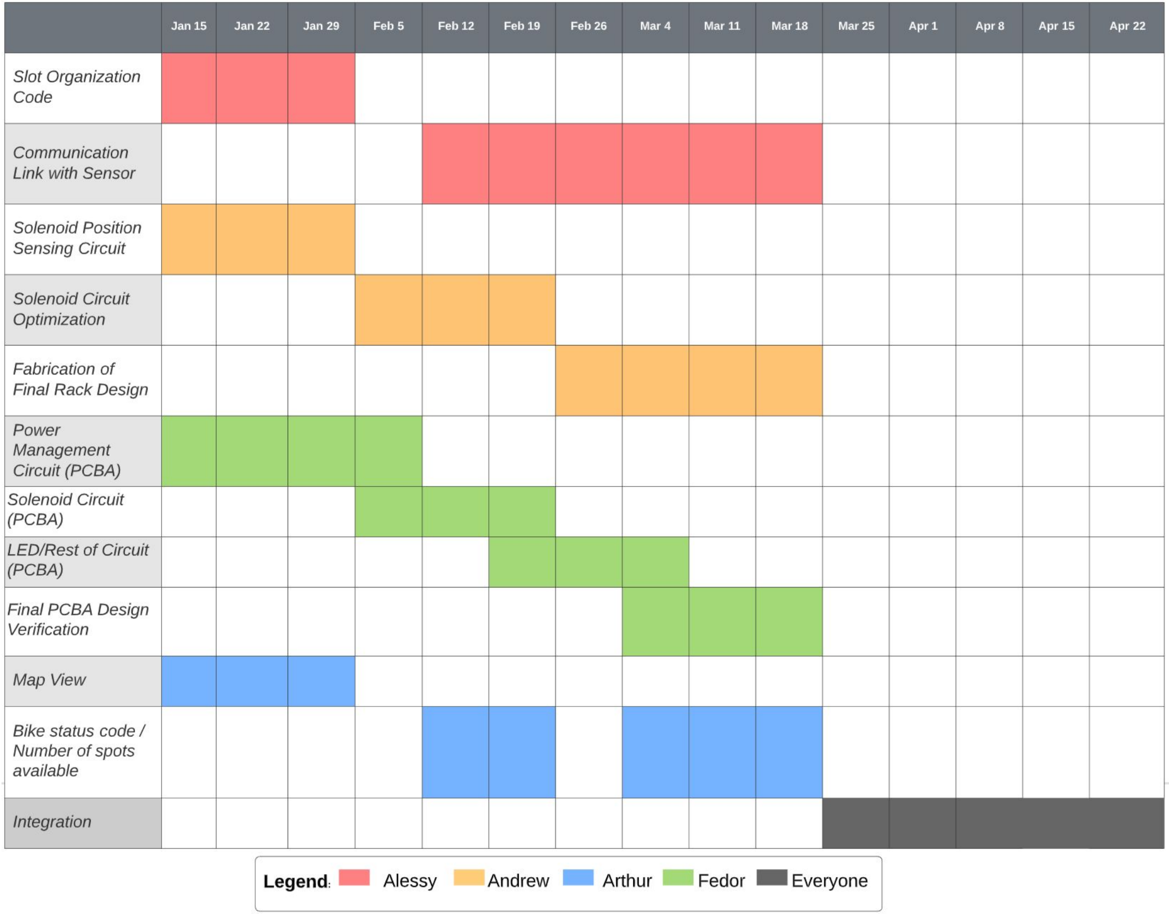
```
/* Establishment of Server that is bound the the local IP of the Raspberry Pi on the current network */
serverSocket = new ServerSocket();
serverSocket.bind(new InetSocketAddress("localhost", 35000));
System.out.println(serverSocket.getInetAddress());
System.out.println("I'm waiting here: " + serverSocket.getLocalPort());
```

Updated Cost

Adafruit HUZZAH ESP-32	21.95
Raspberry Pi kit	69.99
Solenoid (x2)	16.70
RFID Reader and Tag (x2)	10.98
HDMI Cable	6.99
DP to HDMI Cable	8.59
32GB SanDisk microSD	7.64
Raspberry Pi	35
Pi Power Supply	8.49

Total: $186.33

Gantt Chart



	Jan 15	Jan 22	Jan 29	Feb 5	Feb 12	Feb 19	Feb 26	Mar 4	Mar 11	Mar 18	Mar 25	Apr 1	Apr 8	Apr 15	Apr 22
Slot Organization Code	🟥	🟥	🟥												
Communication Link with Sensor					🟥	🟥	🟥	🟥	🟥	🟥					
Solenoid Position Sensing Circuit	🟧	🟧	🟧												
Solenoid Circuit Optimization				🟧	🟧	🟧									
Fabrication of Final Rack Design							🟧	🟧	🟧	🟧					
Power Management Circuit (PCBA)	🟩	🟩	🟩	🟩											
Solenoid Circuit (PCBA)				🟩	🟩	🟩									
LED/Rest of Circuit (PCBA)						🟩	🟩	🟩							
Final PCBA Design Verification								🟩	🟩	🟩					
Map View	🟦	🟦	🟦												
Bike status code / Number of spots available					🟦	🟦		🟦	🟦	🟦					
Integration											⬛	⬛	⬛	⬛	⬛

Legend: 🟥 Alessy 🟧 Andrew 🟦 Arthur 🟩 Fedor ⬛ Everyone

CDR Deliverables - Andrew

- Final model of rack fabricated with metal components (x2)
- Position sensor for solenoid to determine if locking has been successful
- Optimized solenoid driving circuit, focusing on reduction of power consumption

CDR Deliverables - Arthur

- Android Studio Application
  - Two-way transmission between Pi and App
    - Bike rack status
  - Map View
  - Embellish UI

CDR Deliverables - Alessy

- Differentiating between different slots reserved for different users on the SmartRack
- Communication link with sensors to detect if the lock engaged properly
  - Sending that information to the android app in order to let the user know that the lock is properly secured

CDR Deliverables - Fedor

- Final Circuit, with Power Management built out
- Power Management PCBA Circuit designed
  - Solenoid Power
  - Raspberry Pi Power
  - Appropriate power sourced for rest of circuit

Demo

UMassAmherst

The Commonwealth's Flagship Campus