SMART CLOSET

By

Luchuan Zhang

Yiwei Li

Final Report for ECE 445, Senior Design, Spring 2016

TA: Luke Wendt

May, 2016

Project No. 24

Abstract

Our project smart closet is an update to the traditional closet. In the hardware part, it can drive clothes to certain position, light LEDs to instruct users, and take photo for new incoming clothes. In the software aspect, it can store information including worn times and date created for each clothes user stored. The entire system works fine and it can achieve all the features we mentioned in our initial design. There are still some improvements on phone apps and mechanical equipment.

Contents

1. Introduction 1
   1.1 Project Purpose 1
   1.2 Objectives 1
      1.2.1 Goals & Benefits: 1
      1.2.2 Functions & Features: 1
2 Design 2
   2.1 Block Diagrams 2
   2.2 Block Descriptions 2
      2.2.1 Microcontroller (Raspberry Pi) 2
      2.2.2 Distance Sensors (Sharp GP2Y0D810Z0F with Pololu Carrier) 3
      2.2.3 Raspberry Pi HDMI 7” 800*480 Touchscreen 4
      2.2.4 Raspberry Pi Camera Board 4
      2.2.5 LED system 4
      2.2.5 Motor Driver and Motor (Nema 23 Stepper Motor) 5
   2.3 Software Design 6
      2.3.1 Flowchart 6
      2.3.2 Design Description 6
3. Design Verification 8
   3.1 Control system 8
   3.2 User Interface 8
   3.3 Execute Unit 11
   3.4 Data collection Unit 12
4. Costs 13
   4.1 Parts 13
   4.2 Labor 13
5. Conclusion 14
   5.1 Accomplishments 14
   5.2 Uncertainties 14
   5.3 Ethical considerations 14

5.4 Future work 14
References 15
Appendix A 17
Appendix B 22
Appendix C 24

1. Introduction

1.1 Project Purpose

Our project's motivation is inspired by some problems that people met when they trying to organize their clothes in the conventional closet. The first problem is, when clothes are hanging inside the closet, nearby clothes may block the view of one specific clothing item, so users may have trouble picking up the desired clothes. The second problem is some users may have huge closet, and which will cause users to take lots of time to walk to some clothes. The last problem is people sometimes will wear a clothes for several days, but they cannot clearly remember the worn times for each clothes.

We come up with this solution of building a smart system for people's private closet. This closet can show a complete frontal view of each clothes to users in the application, automatically drive the chosen clothes to a center location that is easy for user to pick up, keeps track of every clothes' worn times, and stores created date. In this way, all three problems will be solved in our project.

1.2 Objectives

1.2.1 Goals & Benefits:

- Database for private closet
- Communication between hardware and software
- Help users organize and pick up clothes

1.2.2 Functions & Features:

- Automatically take photos for users' clothes
- Automatically drive chosen clothes to center point of the rack
- Record each clothes' worn times
- Having a touchscreen for Raspberry Pi as user interface
- IR distance sensors to collect physical data
- A conveyor belt system with a stepper motor controlled by Raspberry Pi
- LED system includes 10 LEDs controlled by Raspberry Pi

2 Design

Overall circuit and PCB design is in Appendix B.

2.1 Block Diagrams

Figure 1 is our block diagram, which contains five modules. Red arrows indicate power supply, and black arrows indicate data line.

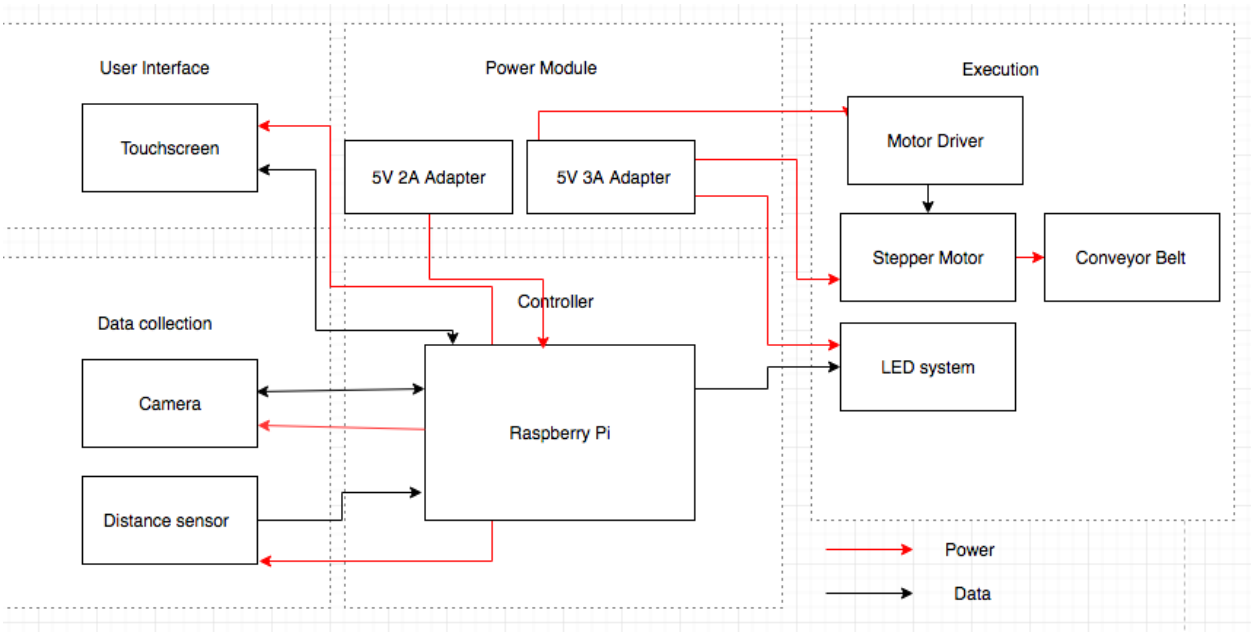


Figure 1 Block diagram

2.2 Block Descriptions

2.2.1 Microcontroller (Raspberry Pi)

Input: 5V micro-USB power supply with 2 A, digital input from one distance sensors, data cable from camera

Output: Control signal to stepper motor driver/stepper motor; control signal to LED system with a level shifter; HDMI signal and power supply to touchscreen

Raspberry Pi is powered by a micro-USB cable with a 5V 2A adapter. It receives digital input from distance sensor through GPIO ports. It connects to the camera board through CSI bus, where the camera board is attached to Raspberry Pi by one of the two small sockets on the board upper surface. Pi gives eight GPIO outputs to motor driver, which drives the stepper motor. Pi links to a touchscreen through USB-micro cable and HDMI cable. The touchscreen can both display and act like a mouse. The control signal to LED system is from the PWM output of Raspberry Pi, which will go through a level shifter to shift from 3.3V to 5V. Table 1 is the pin table for raspberry pi, which shows all connection between the raspberry pi and other devices.

Table 1 Pin table for raspberry pi

Pin	Function	Connection
1	3.3V	Distance sensor
7	GPIO4	To IN1 of motor driver
11	GPIO17	To IN2 of motor driver
12	GPIO18,PCM_clock	Level shifter (LED system)
16	GPIO23	To IN3 of motor driver
18	GPIO24	To IN4 of motor driver
36	GPIO16	To IN5 of motor driver
38	GPIO20	To IN6 of motor driver
40	GPIO21	To IN7 of motor driver
37	GPIO26	To IN8 of motor driver
13	GPIO27	To output of Distance sensor
6	ground	To ground supply
HDMI	HDMI	Touchscreen
Micro USB	Micro USB	Touchscreen
Socket	Socket	Camera Board

2.2.2 Distance Sensors (Sharp GP2Y0D810Z0F with Pololu Carrier)

Input: 5 volts VIN and GND

Output: digital output VOUT links to Raspberry Pi GPIO pin

This is a digital output distance sensor. The output is normally high and when it detects objects in the range from 2cm to 10cm the output will be low. This sensor cannot measure how far the object is away from the sensor but is good at object detecting and counting. Because our project does not need to measure the exact distance between the object and the sensor so we choose this fast distance sensor with a sample rate of almost 400 HZ. [1] Figure 2 is the circuit layout that Pololu carrier has already built for users. So users can simply use the three outputs on the right in figure 2 as ground, vcc and output.

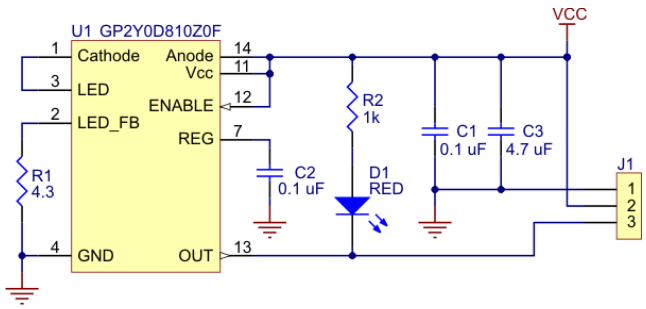


Figure 2 Circuit layout for GP2Y0D810Z0F [3]

2.2.3 Raspberry Pi HDMI 7” 800*480 Touchscreen

Input/Output: HDMI cable from Raspberry Pi, USB cable from Raspberry Pi

This touchscreen drains power from Raspberry Pi through the USB cable, which also roles as a data cable. It will need 600mA if the backlight is fully on. [7] This touchscreen can display the screen of Raspberry Pi and input users' information of finger location to Pi. In our project, this touchscreen will be used as a main user interface, and it will be mounted on the surface of the closet.

2.2.4 Raspberry Pi Camera Board

Input/Output: Flex cable for Raspberry Pi camera

This camera is linked to Raspberry Pi using one of two sockets on the board of Pi. [8] No extra power supply needed. We ordered a 2-meter long flex cable for this camera so it can be attached far enough to get a good photo for the clothes. The camera will input the image to Raspberry Pi and the Pi will control the timing of when to take a photo.

2.2.5 LED system

PCB design for LED system is in Appendix B.

Input: control signal(PWM) from raspberry pi

Output: 10 LEDs

We use 10 NeoPixel RGBW LEDs with integrated driver chips from Adafruit to build up a LED system. This LED chip can display different color to indicate different instructions to users. Each LED can be individually controlled by Raspberry Pi. There will be two PCBs, each contains five LED chips and with a dimension of 25cm*0.5cm. Figure 3 shows the datasheet for neopixel chip, each chip has four output pins. Figure 4 is the circuit diagram for the two LED strips.

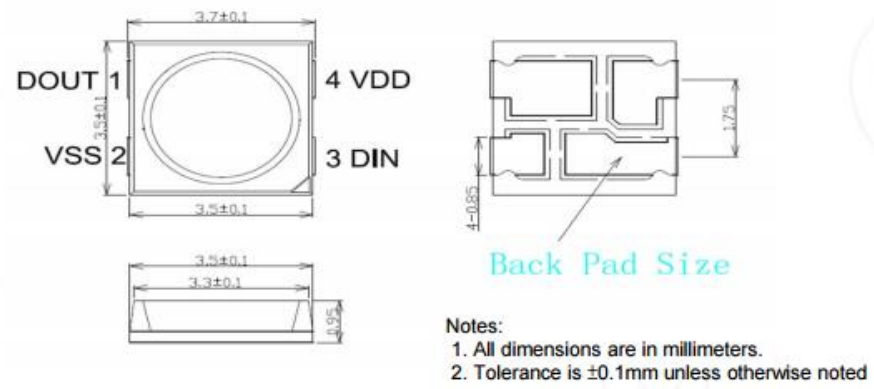


Figure 3 Datasheet for Neopixel [2]

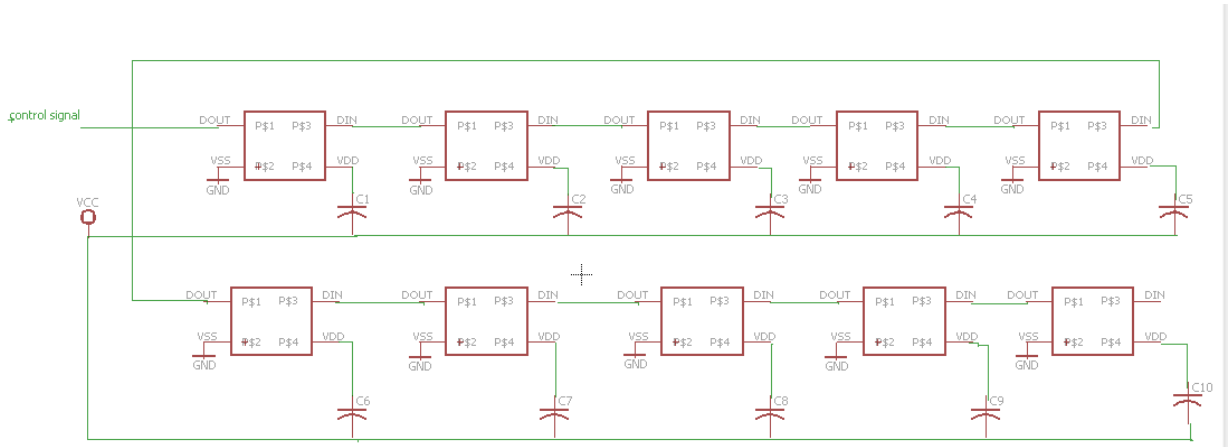


Figure 4 Schematic for LED chip

2.2.5 Motor Driver and Motor (Nema 23 Stepper Motor)

Input: Four control bits, power/ground supply from 5V 3A adapter

Output: Control signal to stepper motor (four wires, black, red, green, blue)

The stepper motor contains two coils [6], which are each driven by two H-bridge circuit. From Figure 5, we can see there are two PNP and two NPN included in each H-bridge. PNP and NPN will behave as a switch. Different combination of input signal will let motor go forward or backward. The rate current for the motor is 1.8 A, and resister for each coil is 2 Ω. Since two transistors are used in series, the voltage drop across the transistors are about 1.4V.

$$I_c = \frac{V_{cc}-1.4}{2} = 1.8A \tag{2.1}$$

From this Equation (2.1), we get Vcc = 5V.

Beta value for the transistors is around 1000. The rate current I_c is 1.8A, $I_b = \frac{I_c}{\beta} = 1.8mA$, a current of 2mA that can be safely drawn from the Raspberry pi.

For NPN transistor,

$$R_b = \frac{3.3-0.7}{0.002} = 1300\Omega \tag{2.2}$$

Figure 5 is the schematic for our motor driver.

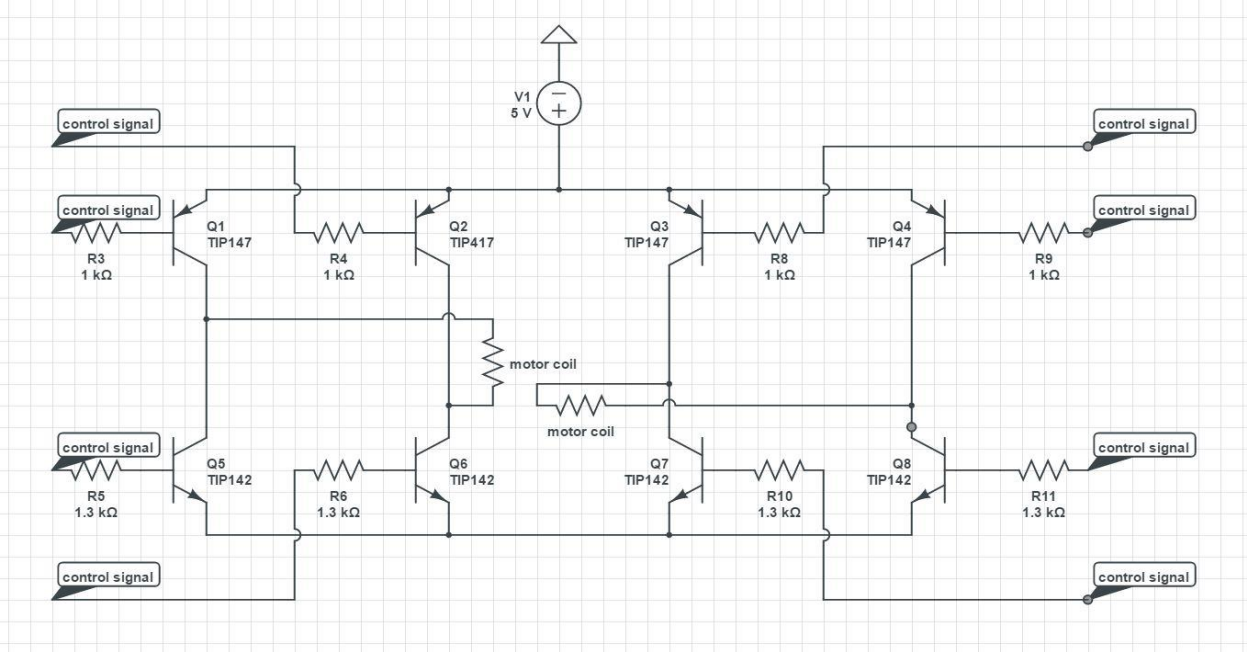


Figure 5 Motor driver schematic [13]

2.3 Software Design

2.3.1 Flowchart

Users can select five functions from the menu, including Pick, Return, Delete, Store, and Laundry. Pick, Return, and Delete are three very similar functions, users just need to select a clothes from the page and confirm the selection to drive the motor and light the LED. For function Store, users should first choose a type from the list, then confirm the choice. After that, the system will take a photo for the incoming clothes and show the photo preview to the user. After the user confirm the photo, a LED will light to indicate the position to store the new clothes. If the user presses the laundry button, all LED that corresponds to clothes which has worn times bigger or equal to two times will be lit up. Figure 6 is the visualization of this description.

2.3.2 Design Description

Software design includes three main functions, one to control motor driver, one to control LED, and the last to construct user interface. The control of motor driver is actually to control eight GPIO outputs of Raspberry pi and send signals in sequence. For the function of controlling LED system, we use the Neopixel library rpi_ws281 [10], which can easily set specific LED to unique color. In the function of user interface, we use the Tkinter library that help us build features such as buttons, radiobuttons, frame, scroll bar and dialogue window. We include the full code in the Appendix C for reference.

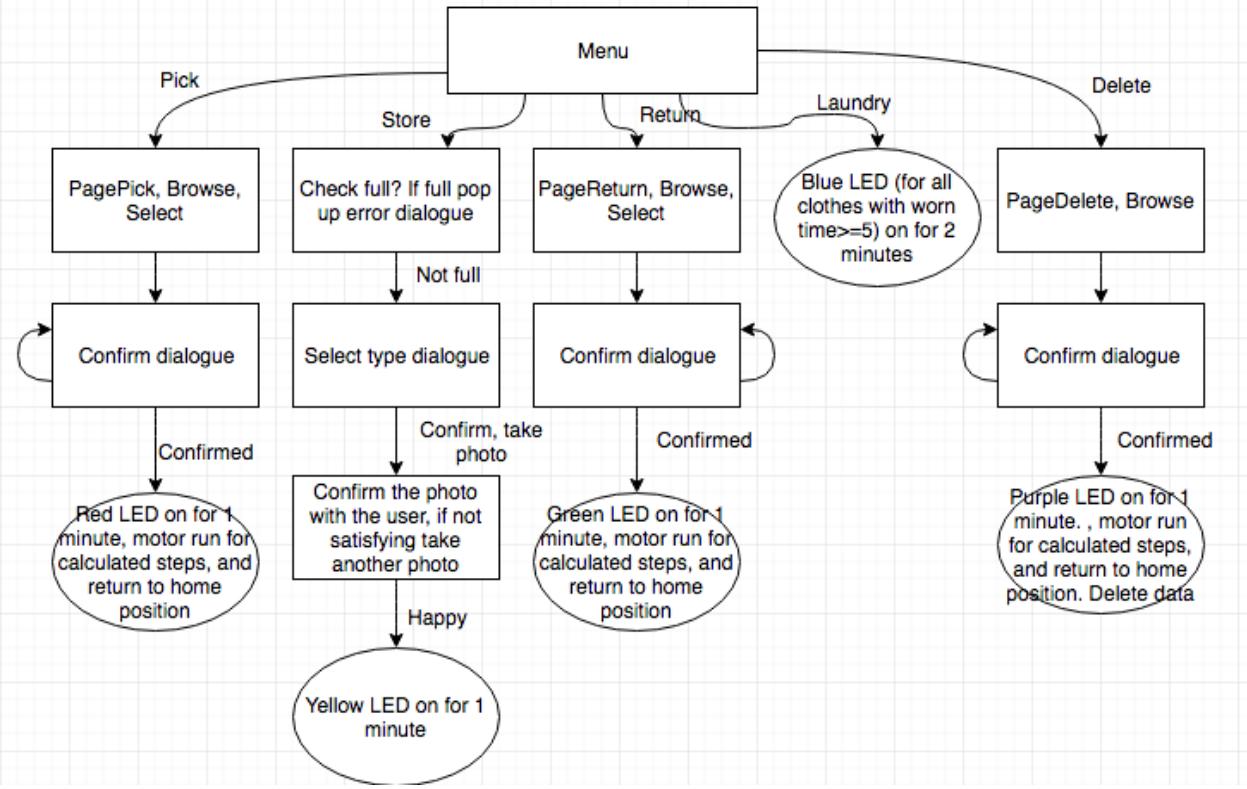


Figure 6 Flowchart

3. Design Verification

We include our R&V table in appendix A.

3.1 Control system

System delay is showed by how soon the motor will stop when an object is detected in front of the distance sensor. It is hard to measure the stop of the motor in the oscilloscope, so we decided to measure the delay by stopwatch on iPhone. The time that recorded by stopwatch is 0.18 seconds, so the delay is perfectly fine with our project design.

3.2 User Interface

Our user interface can run smoothly and follow the flowchart. The only possible flaw is that the Pick, Return, Delete page may not refresh after the user has picked, returned or deleted one clothes. The page will only refresh after the user re-enter the page. Figure 7,8,9,10 shows the function of Store and the background is the home page, where the red button is “Pick”, yellow button is “Store”, green button is “Return”, blue button is “Laundry”, and purple button is “Delete”. Figure 11 shows the Pick page, there are two label blocks in the middle part of the page. The top label block shows information for the clothes on the left, and the bottom label block shows information for the right one. Label block includes information of index, type, worn times, and date created respectively. Other pages, such as return page and delete page look very similar to the pick page.

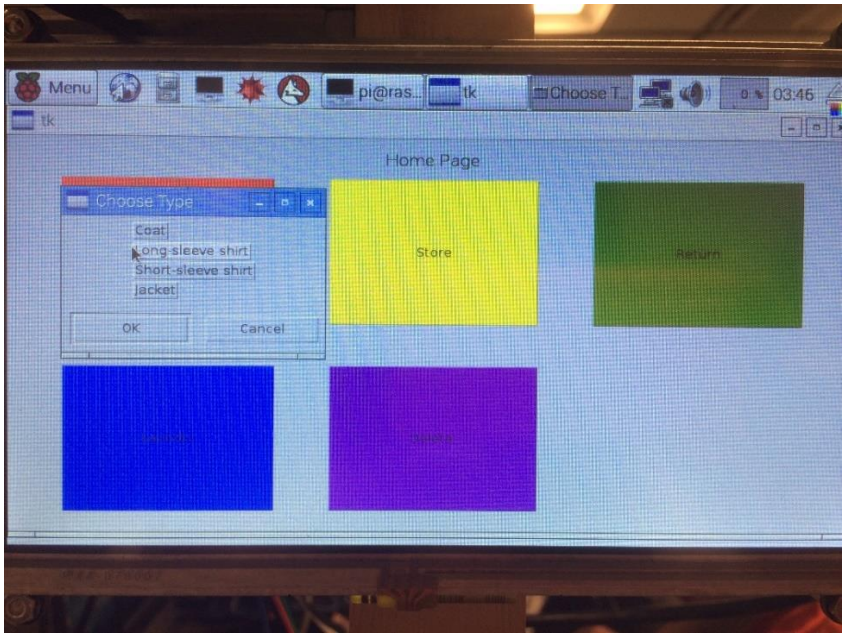


Figure 7 Store function, user choose a type from the list

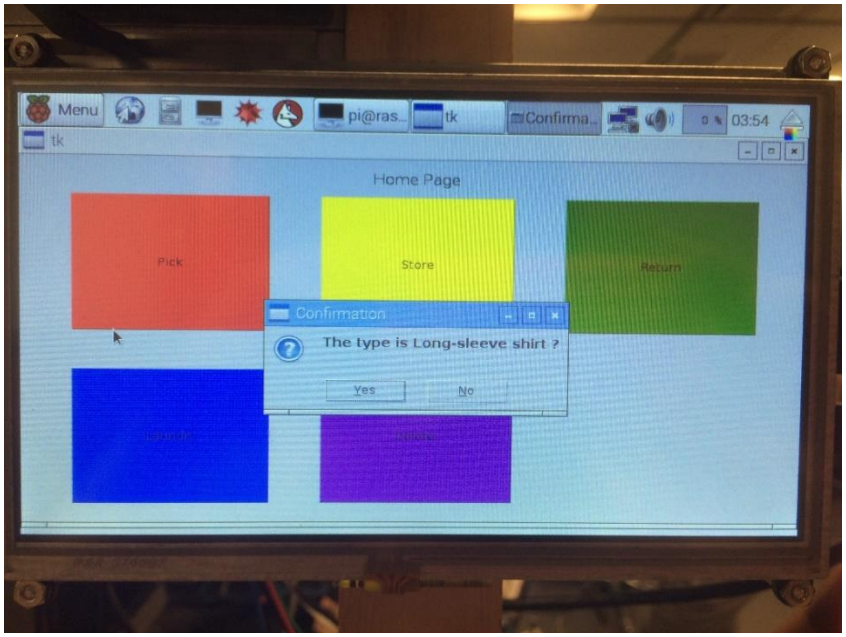


Figure 8 Store function, confirm the type

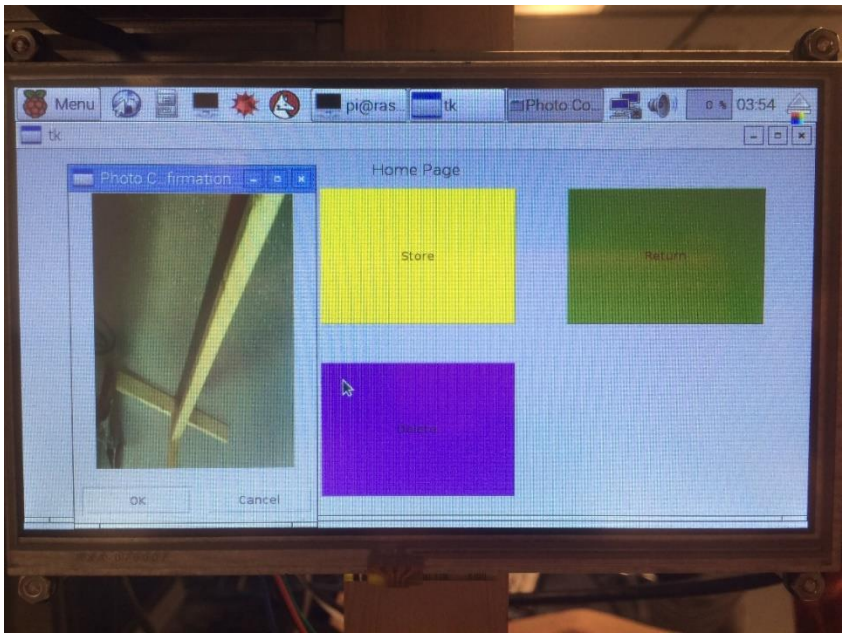


Figure 9 Store function, photo preview

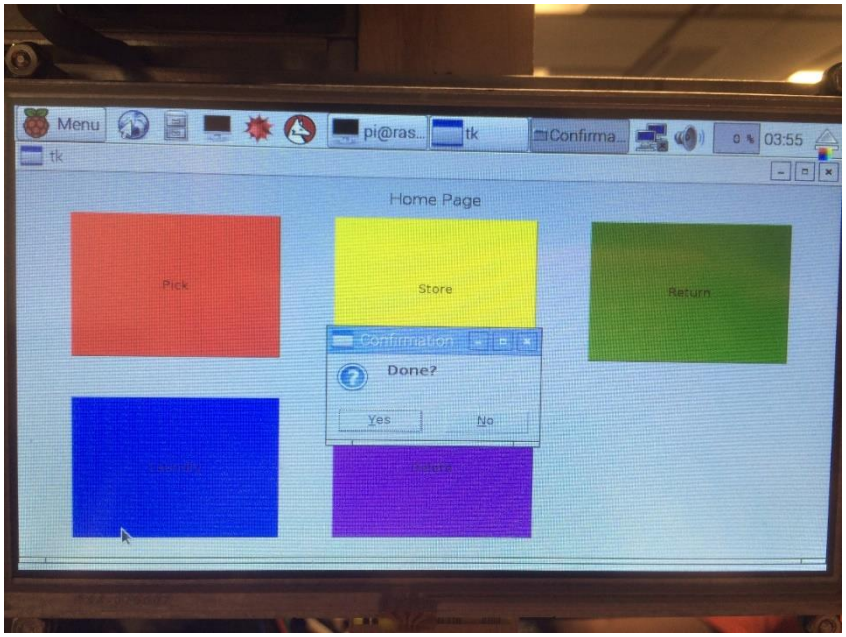


Figure 10 Store function, final confirmation

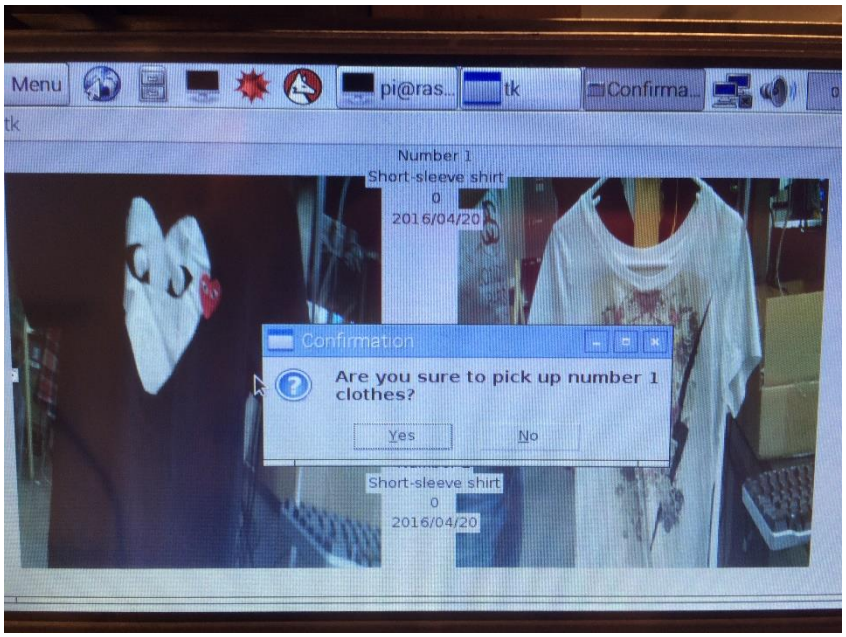


Figure 11 Pick page, confirmation

3.3 Execute Unit

1. Verification for the motor was performed by running the test code in raspberry pi, and the result is motor can rotate forward and backward. In addition to that, the motor can smoothly drive at least 10 clothes on the rack.

2. Verification for the LED system was performed by running the test code in raspberry pi, and the result is LED can light up for correct position and color. Figure 12 shows the lab result.



Figure 12 LED Strip test result

3.4 Data collection Unit

Verification for the distance sensor was performed by attach a voltmeter between output of sensor and ground. Figure 13 shows the result we get from test.

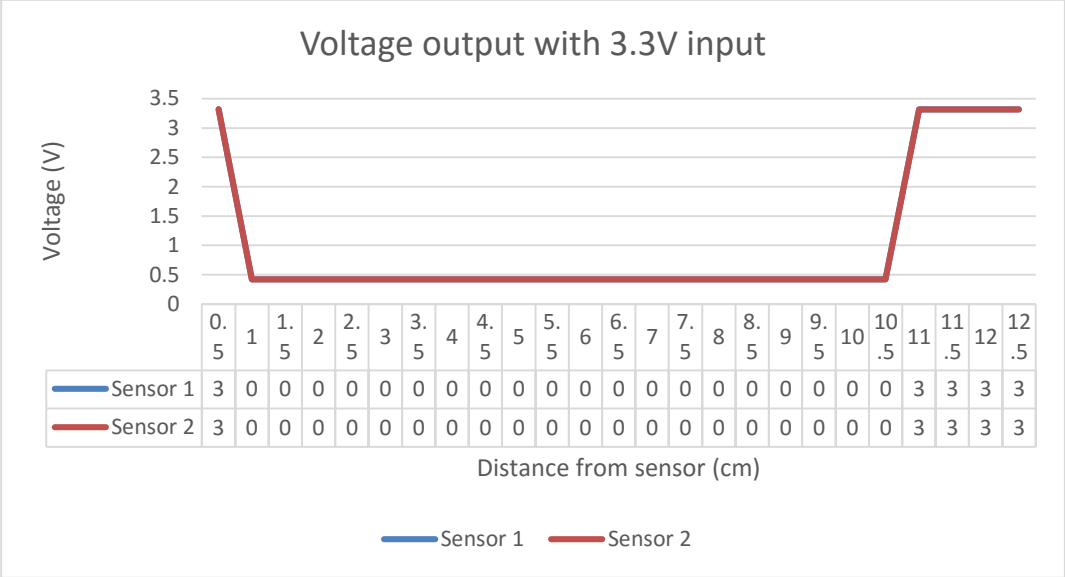


Figure 13 Data for distance sensor

4. Costs

4.1 Parts

Table 2 Parts List

Item	Parts number	Vendor	Quantity	Cost	Actual cost
Distance sensor(infrared Proximity by sharp)	GP2Y0D810Z0F	Adafruit	1	$6.95	$6.95
Raspberry pi	N/A	Adafruit	1	$39.95	$39.95
Neopixel	WS2812	Adafruit	10	$20	$20
Raspberry pi Miniature WIFI	RTI8192	Adafruit	1	$11.95	$11.95
Raspberry pi camera Board	N/A	Adafruit	1	$29.95	$29.95
Stepper motor (1.9Nm)	Nema 23	Amazon	1	$48	$48
Flex cable for camera (2m)	N/A	Adafruit	1	$5.95	$5.95
Raspberry Pi GPIO Breakout board/Cobbler Bundle	N/A	CanaKit	1	$14.99	$14.99
Touchscreen	TFP401	Adafruit	1	$89.95	$89.95
HDMI cable	N/A	Adafruit	1	$4.95	$4.95
USB cable	N/A	Adafruit	2	$5.9	$5.9
Raspberry pi case	N/A	Amazon	1	$10.99	$10.99
5V 3A adaptor	N/A	Amazon	1	$11	$11
Level shifter	74AHCT125	Adafruit	1	$1.5	$1.5
NPN transistor	TIP142	ECE shop	4	$3	$0
PNP transistor	TIP147	ECE shop	4	$3	$0
Resister(1.3k Ω *4, 1.1k Ω *4)	N/A	ECE shop	8	$2	$0
Grand Total				$310.03	$302.03

4.2 Labor

Table 3 Labor cost

Name	Hourly rate	Total hours	Total= hourly Rate *2.5 *total hours
Luchuan zhang	$30	250	$18750
Yiwei li	$30	250	$18750
		500	$37500

5. Conclusion

5.1 Accomplishments

In our project, we construct a user-friendly interface on our 7” touchscreen. User can easily manage and browse their clothes via our program. Our microcontroller can process digit signal input from distance sensor and image data coming from camera. The motor driver design is also match prefect with the stepper motor we have chosen. Our motor can drive at least 10 clothes that satisfy basic requirements for the smart closet. Our LED system can help users quickly locate the correct position for clothes to pick up and position to hang on incoming clothes. The most important accomplishment for our group is that we learn a lot as professional engineer. We learned how to organize the time and workload for our project.

5.2 Uncertainties

When rack is moving, the clothes hang on the rack may sway due to acceleration. The distance between the first clothes to the distance sensor may be affected by this sway. When the user presses Reset button, the closet may not be reset to the original point since distance sensor will detect clothes earlier or late than what we have designed. We think the solution to prevent this case is to hang a much heavy object in front of the first clothes with a low speed, and this object can keep perpendicular to the ground. Our project has set a low speed for motor when resetting the system, but the user should use a heavy clothing item to reset the closet.

5.3 Ethical considerations

It is important to follow the IEEE code of Ethics when we design our project. As a project with motor and convey belt, we need to consider the safety of users. We will add up a protect case for our users that keep users away from the belt and motor. This correlates to the first code of the IEEE Code of Ethics [4]: *1.to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;*

5.4 Future work

1. Build up an ios app, our closet will send all information about owned clothes from Raspberry Pi to users' smartphones via Wi-Fi. Users can easily organize and check the status of all their owned clothes on their smartphones, and label any clothes they want to wear for tomorrow, or any clothes they want to go for laundry or throw away.

2. In the hardware design right now, people need to pick up clothes by themselves, and we want to build up a mechanical device automatically pick up clothes to a basket below the closet.

References

[1] A. Industries, "Sharp GP2Y0D810Z0F digital distance sensor with Pololu carrier [GP2Y0D810Z0F] ". [Online]. Available:

https://www.adafruit.com/products/1927.

[2] NeoPixel Mini 3535 RGB LEDs w/integrated Driver Datasheet [Online]. Available:

https://www.adafruit.com/images/product-files/2686/SK6812MINI_REV.01-1-2.pdf

[3] Datasheet SHARP GP2Y0D810Z0F. [Online]. Available:

http://www.sharp.co.jp/products/device/doc/opto/gp2y0d810z_e.pdf

[4] IEEE Code of Ethics. IEEE. [Online]. Available:

http://www.ieee.org/about/corporate/governance/p7-8.html

[6] Datasheet for Motor (Nema 17 Stepper motor). [Online]. Available:

http://ecx.images-amazon.com/images/I/81RsrEX4IAL.pdf

[7] Touchscreen for Raspberry Pi. [Online]. Available:

https://www.adafruit.com/products/2407

[8] Camera for Raspberry Pi. [Online]. Available:

http://elinux.org/Rpi_Camera_Module

[9] Stepper motor code. [Online]. Available:

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-10-stepper-motors/software

[10] Neopixel software code. [Online]. Available:

https://learn.adafruit.com/neopixels-on-raspberry-pi/software

[11] Vertical scrolled frame code. [Online]. Available:

http://tkinter.unpythonic.net/wiki/VerticalScrolledFrame

[12] Simple dialogue window. [Online]. Available:

http://effbot.org/tkinterbook/tkinter-dialog-windows.htm

[13] "Stepper motor driver,". [Online]. Available:

http://www.hteck.ca/electronics/stepper-motor-drv/sm-driver.html.

[14] Tkinter GUI multiple windows. [Online]. Available:

https://pythonprogramming.net/change-show-new-frame-tkinter/

Appendix A

Table 4 Requirement and Verification Table

Requirements	Verification	Verification status
Control System [5 Points]		
1) Delay between distance sensor and motor feedback should be less than 1 sec	1) (a)Power the Raspberry Pi (b)Program Pi to run the motor (c) Attach oscilloscope in1 to distance sensor output and GND (d) Attach oscilloscope in2 to black and green wire of motor (e) put an object in front of distance sensor (f)measure the time delay between two falling edge in oscilloscope	Y
2) System can control the motor to drive specific clothes that the user chooses to the center of the rack (marked on the rack).	2) (a)open the user interface software (b)select a clothes in the pick module (c)ensure the motor drive the specific clothes to the center of the rack	Y

User Interface [10 Points]		
Graphical User Interface [10] 1) Button in GUI navigate to correct pages to perform correct function	1) (a)Run the application by the command “python app.py” (b) In the home page, pressing “Pick” or “Return” or “Delete”, the app should navigate to a new page with images and labels on it. (c) In the home page, pressing “Store” button, the app should pop up a dialogue window. If the closet is full, the dialogue window is “The closet is full, error.” If not full, the dialogue window shows “The type of this clothes is“, and four options following. (d) In the home page, pressing “Laundry” button, the LED will be on.	Y
2) The dialogue window can pop up at the correct timing	2) (a) After selecting the correct image, a confirmation dialogue window should pop up. (b)If the user press “OK”, the following actions will be made, for example, the LED will on. (c)If the user press “cancel”, the window will disappear, and user will stay at the same page.	Y
3) Images can be selected and can be read by the application	3) (a)First enter “pick” page. (b)Select any picture from the list and press “OK” (c) Ensure number of clothes that console printed match with what you select.	Y

Execute Unit [30 points]		
Motor [10] 1) Can rotate forward and backward (speed bigger than 0.1cm/s) 2) Drive 10 clothes on conveyer belt, clothes weight 0.2kg +- 0.1kg each and dimension is (55cm +-5cm) * (40cm +-5cm)	1) (a)Power Raspberry Pi 5 V 2 A (b)Make sure raspberry pi has installed Linux system on it (Raspbian) (c)Correctly connect the raspberry pi to motor driver (d)Run the code for stepper motor (e)See if motor can rotate forward and backward 2) (a) Put 10 clothes on the belt (b) Run the motor as specified by step 1) (c) Ensure motor have enough torque to drive 10 clothes	Y Y
Motor Driver [10] 1)check switch function for PNP and NPN	1) (a)connect a PNP and a NPN in series (b)add a 5 V power supply between this series circuit (c)input high signal(3.3V) to NPN and low signal(GND) to PNP (d) Attach voltmeter between this series circuit (e) Read voltmeter, ensure it is 5V+-0.5V (f) input low signal to NPN and high signal to PNP (g) Read voltmeter and ensure it is less than 0.4V+-0.2V.	Y

2) there is 1.65V +- 0.1V going through motor coils	2) (a) Connect with Pi and motor using specified pin table (b) input high signal(3.3V) to NPN, and low signal (0,4V) to PNP, let one coil connect between source and ground (c)input high signal(3.3V) to NPN and low signal(0.4V) to PNP (d) Attach voltage meter between this series circuit (e) Read voltage meter, ensure it is 1.65V +- 0.1V	Y
LED System [10] 1)Check if each LED can be controlled individually by Raspberry Pi 2)Each LED can display red, green, yellow, and blue	1) (a)Connect Vcc and GND to NeoPixels PCB (b)Connect GPIO18 to level shifter as shown in the overall schematic diagram (c)Run the testing program on Pi (d)Check each LED can be controlled and lit 2) (a)Complete 1) step (b)Add programs to use four colors on each LED (c)Check each color is correct for every LED on PCB	Y Y
Data Collection [5 Points]		
Camera [2.5] 1) Import picture and video data to Raspberry Pi	1) (a)Connect camera to Pi through socket on board (b)Remotely control Pi through computer, and check the photo/video input	Y

Distance sensor [2.5] 1) Output 0.0-0.6V for object in range from 0 to 10 cm 2) Output 3.3+-0.2V when no object in front of the sensor or object is 11 cm away from sensor	1) (a)Attach voltmeter to Vout of sensor (b)Put clothes in front of the sensor and change the distance from 0 cm to 10 cm (c)Always ensure the reading from voltmeter is in range 0-0.6V 2) (a)Attach voltmeter to Vout of sensor (b)Put clothes in front of the sensor and change the distance from 11 cm to 100 cm (c)Always ensure the reading from voltmeter is in range 3.3V (d)Put nothing in front of the sensor (e)Always ensure the reading from voltmeter is in range 3.3V	Y Y

Appendix B

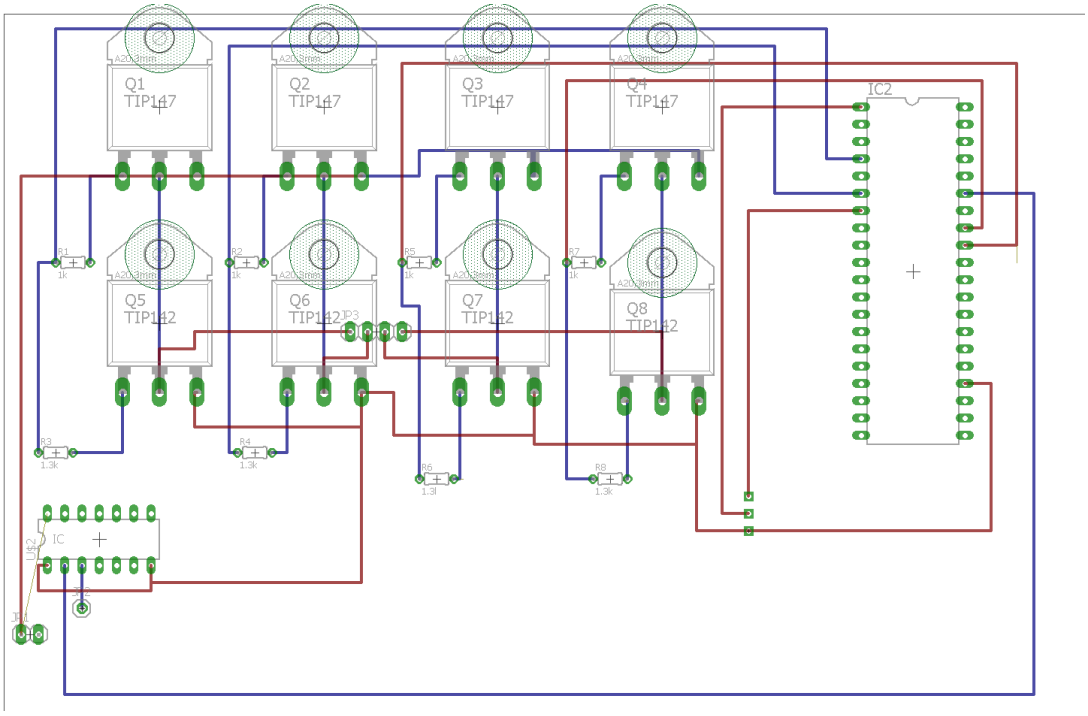


Figure 14 Main PCB schematic

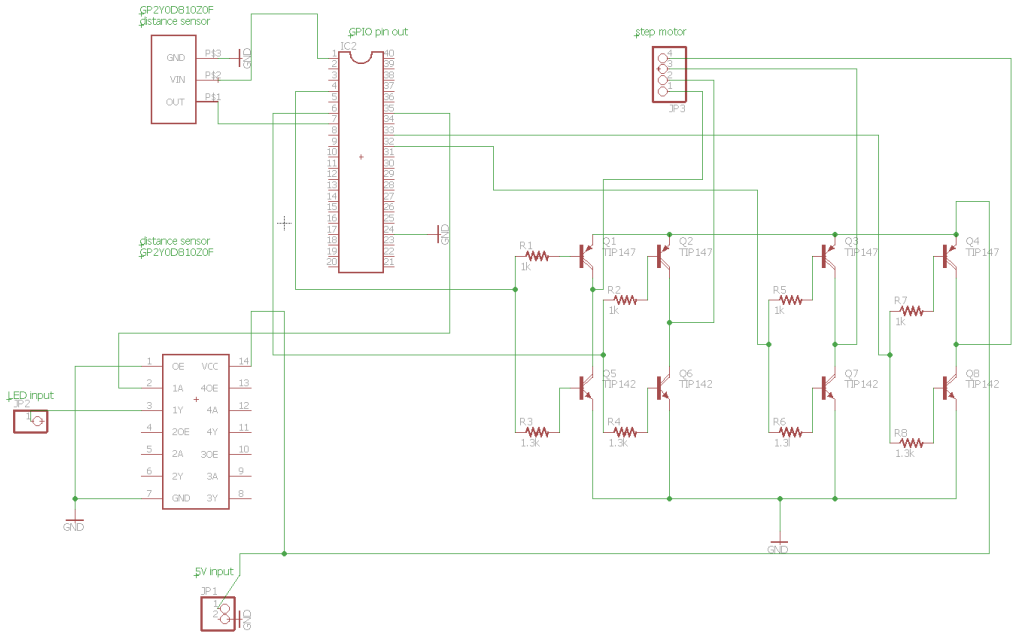


Figure 15 schematic for overall system

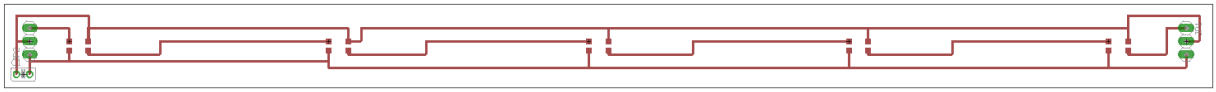


Figure 16 PCB design for LED

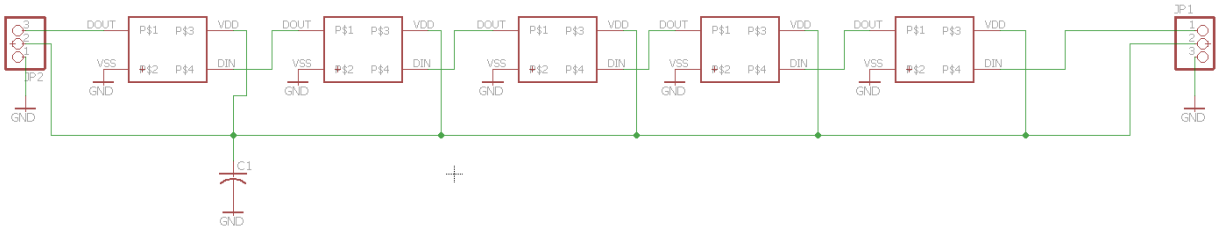


Figure 17 LED circuit

Appendix C

```
# GRB
from Tkinter import *
import tkSimpleDialog
import tkMessageBox
import time
from PIL import ImageTk, Image
from neopixel import *
import picamera
import RPi.GPIO as GPIO
reset = 0
camera = picamera.PiCamera()
LARGE_FONT= ("Verdana", 12)

root = Tk()
root.wm_geometry("800x400")
root.columnconfigure(0, weight = 1)
root.rowconfigure(0,weight = 1)

LED_COUNT      = 10      # Number of LED pixels.                                    ######reference from [10]
LED_PIN        = 18      # GPIO pin connected to the pixels (must support PWM!).
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 5       # DMA channel to use for generating signal (try 5)
LED_BRIGHTNESS = 200     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)

strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT,
LED_BRIGHTNESS)
strip.begin()

GPIO.setmode(GPIO.BCM)

coil_A_1_pin = 4                                                                  #####reference from [9]
coil_A_1_pin_c = 12
coil_A_2_pin = 17
coil_A_2_pin_c = 26
coil_B_1_pin = 23
coil_B_1_pin_c = 20
coil_B_2_pin = 24
coil_B_2_pin_c = 21
sensor1 = 27

GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_1_pin_c, GPIO.OUT)

GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin_c,GPIO.OUT)
```

```
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin_c,GPIO.OUT)

GPIO.setup(coil_B_2_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin_c,GPIO.OUT)

GPIO.setup(sensor1, GPIO.IN)
sensor2 = 1
def forward(steps):
   delay = int(10)/1000.0
   for i in range(0, steps):
        if GPIO.input(sensor1) == 0:
        print "break"
        break
        else:
        setStep(1, 0, 1, 0)
        time.sleep(delay)
        setStep(0, 1, 1, 0)
        time.sleep(delay)
        setStep(0, 1, 0, 1)
        time.sleep(delay)
        setStep(1, 0, 0, 1)
        time.sleep(delay)
def forwardreset(steps):
   delay = int(50)/1000.0
   for i in range(0, steps):
        if GPIO.input(sensor1) == 0:
        print "break"
        break
        else:
        setStep(1, 0, 1, 0)
        time.sleep(delay)
        setStep(0, 1, 1, 0)
        time.sleep(delay)
        setStep(0, 1, 0, 1)
        time.sleep(delay)
        setStep(1, 0, 0, 1)
        time.sleep(delay)


def backwards(steps):
   delay = int(10)/1000.0
   for i in range(0, steps):
        if sensor2 == 0:
        print "break"
        break
        else:
```

```
        setStep(1, 0, 0, 1)
        time.sleep(delay)
        setStep(0, 1, 0, 1)
        time.sleep(delay)
        setStep(0, 1, 1, 0)
        time.sleep(delay)
        setStep(1, 0, 1, 0)
        time.sleep(delay)


def setStep(w1, w2, w3, w4):
    GPIO.output(coil_A_1_pin, w1)
    GPIO.output(coil_A_1_pin_c,w1)
    GPIO.output(coil_A_2_pin, w2)
    GPIO.output(coil_A_2_pin_c,w2)
    GPIO.output(coil_B_1_pin, w3)
    GPIO.output(coil_B_1_pin_c, w3)
    GPIO.output(coil_B_2_pin, w4)
    GPIO.output(coil_B_2_pin_c,w4)


class Clothes:

    def __init__(self):
        self.type = ["none","none","none","none","none","none","none","none","none","none"]
        self.worn_time = [0,0,0,0,0,0,0,0,0,0]
        self.date_created = ["0","0","0","0","0","0","0","0","0","0"]
        self.image_name = ["0.jpg", "1.jpg","2.jpg","3.jpg", "4.jpg", "5.jpg", "6.jpg","7.jpg","8.jpg","9.jpg"]
        self.status = [0,0,0,0,0,0,0,0,0,0]
        self.permanent = [0,0,0,0,0,0,0,0,0,0]

    def get_current_number(self):
        num = 0
        for i in self.status:
            if(i == 1):
                num = num +1
        return num

    def get_permanent_number(self):
        num = 0
        for i in self.permanent:
            if(i == 1):
                num = num +1
        return num

    def get_return_number(self):
        num = 0
        for idx,val in enumerate(self.permanent):
```

```
        if(val == 1 and self.status[idx] == 0):
            num = num +1
    return num

def get_return_list(self):
    list = []
    for idx,val in enumerate(self.permanent):
        if(val == 1 and self.status[idx] == 0):
            list.append(idx)
    return list

def add_clothes(self, index, type, date):
    if((index-1)>=0 and (index-1)<=9):
        self.type[index-1] = type
        self.date_created[index-1] = date
        self.status[index-1] = 1
        self.permanent[index-1] = 1
          ##print "added 1 clothes index is %i" % (index-1)


def pick_clothes(self, index):
    if(index>=0 and index<=9):
        self.status[index] = 0
        self.worn_time[index] = self.worn_time[index]+1

def return_clothes(self,index):
    if(index>=0 and index<=9):
        self.status[index] = 1

def delete_clothes(self,index):
    if(index>=0 and index<=9):
        self.status[index] = 0
        self.permanent[index] = 0


def assign_index(self,type):
    emptyslots = [i for i, e in enumerate(self.permanent) if e == 0]
      print(emptyslots)
      index = emptyslots[0]

    #if(type == "Long-sleeve shirt"):
     #   for i in emptyslots:
      #      if(i >= 2):
       #         index = i
        #        break
    #if(type == "Short-sleeve shirt"):
     #   for i in emptyslots:
      #      if(i >= 4):
```

```
            #        index = i
             #       break
        #if(type == "Jacket"):
         #   for i in emptyslots:
          #     if(i >= 6):
            #        index = i
             #       break
        print "index is %i" % index
        return index

class VerticalScrolledFrame(Frame):                                    #####reference from [11]
    """A pure Tkinter scrollable frame that actually works!
       * Use the 'interior' attribute to place widgets inside the scrollable frame
       * Construct and pack/place/grid normally
       * This frame only allows vertical scrolling

       """
    def __init__(self, parent, *args, **kw):
        Frame.__init__(self, parent, *args, **kw)

        # create a canvas object and a vertical scrollbar for scrolling it
        vscrollbar = Scrollbar(self, orient=VERTICAL, width = 50)
        vscrollbar.pack(fill=Y, side=RIGHT, expand=TRUE)
        canvas = Canvas(self, bd=0, highlightthickness=0,yscrollcommand=vscrollbar.set)
        canvas.pack(side=LEFT, fill=BOTH, expand=TRUE)
        vscrollbar.config(command=canvas.yview)

        # reset the view
        canvas.xview_moveto(0)
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled with it
        self.interior = interior = Frame(canvas)
        interior_id = canvas.create_window(0, 0, window=interior,anchor=NW)

        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner frame
            size = (800, 2000)
            canvas.config(scrollregion="0 0 %s %s" % size)
            if 800 != canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=800)
            if 400 != canvas.winfo_height():
                # update the canvas's height to fit the inner frame
                canvas.config(height=400)
```

```
        interior.bind('<Configure>', _configure_interior)

        def _configure_canvas(event):
            if 800 != canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id, width=canvas.winfo_width())
            if 400 != canvas.winfo_height():
                # update the inner frame's height to fill the canvas
                canvas.itemconfigure(interior_id, height=canvas.winfo_height())

        canvas.bind('<Configure>', _configure_canvas)

class ConfirmPhoto(tkSimpleDialog.Dialog):                    ####reference from [12]

    def body(self, master):
        self.index = self.input-1
        imgin = Image.open(str(self.index)+".jpg")
        imgin = imgin.resize((200,300), Image.ANTIALIAS)
        cimage1 = ImageTk.PhotoImage(imgin)
        label1 = Label(self, image = cimage1)
        label1.photo = cimage1
        label1.pack()



    def apply(self):
        clothes.add_clothes(self.input, self.input2, self.input3)
        num = clothes.get_current_number()
          print(num)
          strip.setPixelColorRGB((self.input-1),69,255,0)
        strip.show()
        #yellow led on(done)
        if tkMessageBox.askyesno("Confirmation","Done?"):
            #led off(done)
            strip.setPixelColorRGB((self.input-1), 0,0,0)
            strip.show()
            print "yellow led off"

            #wait 1 minute, led off(done)
        else:
              time.sleep(30)
            strip.setPixelColorRGB((self.input-1), 0,0,0)
            strip.show()


class TypeChoice(tkSimpleDialog.Dialog):

    def body(self, master):
```

```
        self.var = IntVar()
        b1 = Radiobutton(master, indicatoron = 0, text = "Coat", variable = self.var, value = 1)
        b1.pack(anchor=W)
        b2 = Radiobutton(master, indicatoron = 0, text = "Long-sleeve shirt", variable = self.var, value = 2)
        b2.pack(anchor=W)
        b3 = Radiobutton(master, indicatoron = 0, text = "Short-sleeve shirt", variable = self.var, value = 3)
        b3.pack(anchor=W)
        b4 = Radiobutton(master, indicatoron = 0, text = "Jacket", variable = self.var, value = 4)
        b4.pack(anchor=W)


    def apply(self):
        if(self.var.get()):
            if(self.var.get() == 1):
                type = "Coat"
            if(self.var.get() == 2):
                type = "Long-sleeve shirt"
            if(self.var.get() == 3):
                type = "Short-sleeve shirt"
            if(self.var.get() == 4):
                type = "Jacket"
            if tkMessageBox.askyesno("Confirmation", "The type is %s ?" % type):
                    print type
                    index = clothes.assign_index(type)
                    camera.capture(clothes.image_name[index])
                   #take photo(done)
                    #save photo(done)

                    d = ConfirmPhoto(root,title = "Photo Confirmation", input = (index+1), input2 = type, input3 =
time.strftime("%Y/%m/%d"))



def StorePressed():
    if (clothes.get_permanent_number() < 10):
        d = TypeChoice(root, title= "Choose Type")
    else:
        tkMessageBox.showerror("Error", "The closet is full.")

def ResetPressed():
    forwardreset(1000)
    setStep(0,0,0,0)

def LaundryPressed():
    #light on(done)
    laundry = []
    for i in range (0, 10):
```

```
        if clothes.worn_time[i] >= 2 and clothes.status[i] == 1:
            strip.setPixelColorRGB(i,0,0,255)
            strip.show()
            laundry.append(i)

    if tkMessageBox.askyesno("Confirmation", "Have picked all of them?"):
        #light off(done)
        for index,item in enumerate (laundry):
            strip.setPixelColorRGB(item, 0,0,0)
            strip.show()
              clothes.status[item] = 0
              clothes.worn_time[item] = 0
            print "done"
    else:
        #after one minute light off(done)
        time.sleep(30)
        for index,item in enumerate (laundry):
            strip.setPixelColorRGB(item,0,0,0)
            strip.show()



class main(Frame):                              ####reference from [14]

    def __init__(self, *args, **kwargs):

        Frame.__init__(self, *args, **kwargs)
        self.container = Frame(self, width = 800, height = 480)

        self.container.pack(side = "top", fill="both", expand = True)

        self.container.grid_rowconfigure(0, weight=1)
        self.container.grid_columnconfigure(0, weight=1)
        self.frames = {}

        for F in (StartPage, PagePick, PageReturn, PageDelete):

          frame = F(self.container, self)

          self.frames[F] = frame

          frame.grid(row=0, column=0, sticky="nsew")


          frame = self.frames[StartPage]
          frame.tkraise()

    def show_frame(self, cont,*args, **kwargs):
```

```
        self.container.destroy()
      self.container = Frame(self, width = 800, height = 480)
      self.container.pack(side = "top", fill="both", expand = True)
      self.container.grid_rowconfigure(0, weight=1)
      self.container.grid_columnconfigure(0, weight=1)

      frame = cont(self.container, self)
        self.frames[cont] = frame
      frame.grid(row=0, column=0, sticky="nsew")
      frame.tkraise()


class StartPage(Frame):

   def __init__(self, parent, controller):
      Frame.__init__(self,parent)
      label = Label(self, text="Home Page", font=LARGE_FONT)
      label.pack(pady=10,padx=10)

      PickButton = Button(self, text="Pick", command = lambda: controller.show_frame(PagePick), bg =
"red")
      PickButton.place(x = 50, y = 40, height = 150, width = 200)
      StoreButton = Button(self, text="Store", command = StorePressed, bg = "yellow")
      StoreButton.place(x = 300, y = 40, height = 150, width = 200)
      ReturnButton = Button(self, text="Return", command = lambda:
controller.show_frame(PageReturn), bg = "green")
      ReturnButton.place(x = 550, y = 40, height = 150, width = 200)
      LaundryButton = Button(self, text="Laundry", command = LaundryPressed, bg = "blue")
      LaundryButton.place(x = 50, y = 230, height = 150, width = 200)

      DeleteButton = Button(self, text="Delete", command = lambda:
controller.show_frame(PageDelete), bg = "purple")
      DeleteButton.place(x = 300, y = 230, height = 150, width = 200)

      ResetButton = Button(self, text = "Reset", command = ResetPressed, bg = "white")
      ResetButton.place(x = 550, y = 230, height = 150, width = 200)



picked_index = IntVar()
class PagePick(Frame):

   def PickSelected(*args):
      num = picked_index.get()
      print "PickSelected"
      if tkMessageBox.askyesno("Confirmation", "Are you sure to pick up number %i clothes?" % (num +
1)):
         clothes.pick_clothes(num)
```

```
#red light on(done)
#motor move and return to home position
if(num == 0):
            backwards(140)
  elif(num == 1):
  backwards(120)
  elif(num == 2):
  backwards(100)
  elif(num == 3):
  backwards(80)
  elif(num == 4):
  backwards(60)
  elif(num == 5):
  backwards(50)
  elif(num == 6):
  backwards(70)
  elif(num == 7):
  backwards(90)
  elif(num == 8):
  backwards(110)
  elif(num == 9):
  backwards(130)
setStep(0,0,0,0)
  time.sleep(1)
strip.setPixelColorRGB(num,0,255,0)
  strip.show()
#GPIO.cleanup()
if tkMessageBox.askyesno("Confirmation", "Have picked?"):
   #red light off(done)
  if(num == 0):
            forward(140)
  elif(num == 1):
            forward(120)
  elif(num == 2):
            forward(100)
  elif(num == 3):
            forward(80)
  elif(num == 4):
            forward(60)
  elif(num == 5):
            forward(50)
  elif(num == 6):
            forward(70)
  elif(num == 7):
            forward(90)
  elif(num == 8):
            forward(110)
  elif(num == 9):
```

```
                    forward(130)
            setStep(0,0,0,0)
           time.sleep(1)
            strip.setPixelColorRGB(num, 0,0,0)
           strip.show()
           print "done"
         else:
            #after one minute light off(done)
            time.sleep(30)
            strip.setPixelColorRGB(num,0,0,0)
            strip.show()


  def __init__(self, parent, controller):
     Frame.__init__(self, parent)
     self.verticalframe = VerticalScrolledFrame(self)
     self.verticalframe.pack()
     labels = []


     NumOfLabels = 0
     c_num = clothes.get_current_number()
     print "c_num = %i" % c_num

     if c_num == 1 or c_num == 2:
        NumOfLabels = 22
     if c_num == 3 or c_num == 4:
        NumOfLabels = 44
     if c_num == 5 or c_num == 6:
        NumOfLabels = 66
     if c_num == 7 or c_num == 8:
        NumOfLabels = 88
     if c_num == 9 or c_num == 10:
        NumOfLabels = 110

     occupied_slots = [i for i, e in enumerate(clothes.status) if e != 0]
     positionx = [30,390,30,390,30,390,30,390,30,390]
     positiony = [25,25,425,425,825,825,1225,1225,1625,1625]
     currentclothes = []
     for idx,val in enumerate(occupied_slots):
        currentclothes.append(val)
        imgin = Image.open(str(val)+".jpg")
        imgin = imgin.resize((300,400), Image.ANTIALIAS)
        cimage = ImageTk.PhotoImage(imgin)
        b = Radiobutton(self.verticalframe.interior, image = cimage, variable = picked_index, value =
val,command = lambda:self.PickSelected())
        b.image = cimage
        b.place(x = positionx[idx], y = positiony[idx], height = 350, width = 300)
```

```
print "pick page"

for i in range(NumOfLabels):
   if (i == 0):
     index = "Number "+str(currentclothes[0]+1)
     labels.append(Label(self.verticalframe.interior, text = index))
     elif (i == 1):
     type = clothes.type[currentclothes[0]]
     labels.append(Label(self.verticalframe.interior, text = type))
     elif (i == 2):
     worntime = clothes.worn_time[currentclothes[0]]
     labels.append(Label(self.verticalframe.interior, text = str(worntime)))
     elif (i == 3):
     date = clothes.date_created[currentclothes[0]]
     labels.append(Label(self.verticalframe.interior, text = date))
   elif (i == 15 and (len(currentclothes) >= 2)):
      index = "Number "+str(currentclothes[1]+1)
     labels.append(Label(self.verticalframe.interior, text = index))
     elif (i == 16 and (len(currentclothes) >= 2)):
  type = clothes.type[currentclothes[1]]
  labels.append(Label(self.verticalframe.interior, text = type))
   elif (i == 17 and (len(currentclothes) >= 2)):
  worntime = clothes.worn_time[currentclothes[1]]
  labels.append(Label(self.verticalframe.interior, text = str(worntime)))
   elif (i == 18 and (len(currentclothes) >= 2)):
  date = clothes.date_created[currentclothes[1]]
  labels.append(Label(self.verticalframe.interior, text = date))
   elif (i == 23):
     index = "Number "+str(currentclothes[2]+1)
     labels.append(Label(self.verticalframe.interior, text = index))
     elif (i == 24):
  type = clothes.type[currentclothes[2]]
  labels.append(Label(self.verticalframe.interior, text = type))
   elif (i == 25):
  worntime = clothes.worn_time[currentclothes[2]]
  labels.append(Label(self.verticalframe.interior, text = str(worntime)))
   elif (i == 26):
  date = clothes.date_created[currentclothes[2]]
  labels.append(Label(self.verticalframe.interior, text = date))
   elif (i == 36 and (len(currentclothes) >= 4)):
      index = "Number "+str(currentclothes[3]+1)
     labels.append(Label(self.verticalframe.interior, text = index))
   elif (i == 37 and (len(currentclothes) >= 4)):
  type = clothes.type[currentclothes[3]]
  labels.append(Label(self.verticalframe.interior, text = type))
   elif (i == 38 and (len(currentclothes) >= 4)):
  worntime = clothes.worn_time[currentclothes[3]]
  labels.append(Label(self.verticalframe.interior, text = str(worntime)))
```

```
 elif (i == 39 and (len(currentclothes) >= 4)):
date = clothes.date_created[currentclothes[3]]
labels.append(Label(self.verticalframe.interior, text = date))
   elif (i == 45):
   index = "Number "+str(currentclothes[4]+1)
labels.append(Label(self.verticalframe.interior, text = index))
 elif (i == 46):
type = clothes.type[currentclothes[4]]
labels.append(Label(self.verticalframe.interior, text = type))
 elif (i == 47):
worntime = clothes.worn_time[currentclothes[4]]
labels.append(Label(self.verticalframe.interior, text = str(worntime)))
 elif (i == 48):
date = clothes.date_created[currentclothes[4]]
labels.append(Label(self.verticalframe.interior, text = date))
 elif (i == 58 and (len(currentclothes) >= 6)):
    index = "Number "+str(currentclothes[5]+1)
    labels.append(Label(self.verticalframe.interior, text = index))
 elif (i == 59 and (len(currentclothes) >= 6)):
type = clothes.type[currentclothes[5]]
labels.append(Label(self.verticalframe.interior, text = type))
 elif (i == 60 and (len(currentclothes) >= 6)):
worntime = clothes.worn_time[currentclothes[5]]
labels.append(Label(self.verticalframe.interior, text = str(worntime)))
 elif (i == 61 and (len(currentclothes) >= 6)):
date = clothes.date_created[currentclothes[5]]
labels.append(Label(self.verticalframe.interior, text = date))
 elif (i == 67):
    index = "Number "+str(currentclothes[6]+1)
    labels.append(Label(self.verticalframe.interior, text = index))
 elif (i == 68):
type = clothes.type[currentclothes[6]]
labels.append(Label(self.verticalframe.interior, text = type))
 elif (i == 69):
worntime = clothes.worn_time[currentclothes[6]]
labels.append(Label(self.verticalframe.interior, text = str(worntime)))
 elif (i == 70):
date = clothes.date_created[currentclothes[6]]
labels.append(Label(self.verticalframe.interior, text = date))
 elif (i == 80 and (len(currentclothes) >= 8)):
    index = "Number "+str(currentclothes[7]+1)
    labels.append(Label(self.verticalframe.interior, text = index))
 elif (i == 81 and (len(currentclothes) >= 8)):
type = clothes.type[currentclothes[7]]
labels.append(Label(self.verticalframe.interior, text = type))
 elif (i == 82 and (len(currentclothes) >= 8)):
worntime = clothes.worn_time[currentclothes[7]]
labels.append(Label(self.verticalframe.interior, text = str(worntime)))
```

```
            elif (i == 83 and (len(currentclothes) >= 8)):
            date = clothes.date_created[currentclothes[7]]
            labels.append(Label(self.verticalframe.interior, text = date))
            elif (i == 89):
                index = "Number "+str(currentclothes[8]+1)
                labels.append(Label(self.verticalframe.interior, text = index))
            elif (i == 90):
            type = clothes.type[currentclothes[8]]
            labels.append(Label(self.verticalframe.interior, text = type))
            elif (i == 91):
            worntime = clothes.worn_time[currentclothes[8]]
            labels.append(Label(self.verticalframe.interior, text = str(worntime)))
            elif (i == 92):
            date = clothes.date_created[currentclothes[8]]
            labels.append(Label(self.verticalframe.interior, text = date))
            elif (i == 102 and (len(currentclothes) >= 10)):
                index = "Number "+str(currentclothes[9]+1)
                labels.append(Label(self.verticalframe.interior, text = index))
            elif (i == 103 and (len(currentclothes) >= 10)):
            type = clothes.type[currentclothes[9]]
            labels.append(Label(self.verticalframe.interior, text = type))
            elif (i == 104 and (len(currentclothes) >= 10)):
            worntime = clothes.worn_time[currentclothes[9]]
            labels.append(Label(self.verticalframe.interior, text = str(worntime)))
            elif (i == 105 and (len(currentclothes) >= 10)):
            date = clothes.date_created[currentclothes[9]]
            labels.append(Label(self.verticalframe.interior, text = date))
                else:
                        labels.append(Label(self.verticalframe.interior, text=""))
            labels[i].pack()

        #picked_index.trace("w", self.PickSelected)

        HomeButton = Button(self.verticalframe.interior, text = "Home Page", command=lambda:
controller.show_frame(StartPage) )
        HomeButton.pack()




return_index = IntVar()
class PageReturn(Frame):

    def ReturnSelected(*args):
        num = return_index.get()
        if tkMessageBox.askyesno("Confirmation", "Are you sure to return number %i clothes?" % (num +
1)):
            clothes.return_clothes(num)
```

```
if(num == 0):
   backwards(140)
elif(num == 1):
   backwards(120)
elif(num == 2):
   backwards(100)
elif(num == 3):
   backwards(80)
elif(num == 4):
   backwards(60)
elif(num == 5):
   backwards(50)
elif(num == 6):
   backwards(70)
elif(num == 7):
   backwards(90)
elif(num == 8):
   backwards(110)
elif(num == 9):
   backwards(130)
setStep(0,0,0,0)
time.sleep(1)
strip.setPixelColorRGB(num,255,0,0)
strip.show()
#green light on(done)
if tkMessageBox.askyesno("Confirmation", "Have picked?"):
   #green light off(done)
   if(num == 0):
      forward(140)
   elif(num == 1):
      forward(120)
   elif(num == 2):
      forward(100)
   elif(num == 3):
      forward(80)
   elif(num == 4):
      forward(60)
   elif(num == 5):
      forward(50)
   elif(num == 6):
      forward(70)
   elif(num == 7):
      forward(90)
   elif(num == 8):
      forward(110)
   elif(num == 9):
      forward(130)
   setStep(0,0,0,0)
```

```
            time.sleep(1)
            strip.setPixelColorRGB(num, 0,0,0)
            strip.show()
            print "done"
        else:
            #after one minute light off(done)
            time.sleep(30)
            strip.setPixelColorRGB(num,0,0,0)
            strip.show()


    def __init__(self, parent, controller):
        Frame.__init__(self, parent)
        self.verticalframe = VerticalScrolledFrame(self)
        self.verticalframe.pack()
        labels = []

        NumOfLabels = 0
        r_num = clothes.get_return_number()
        print "r_num = %i" % r_num
        if r_num == 1 or r_num == 2:
            NumOfLabels = 22
        if r_num == 3 or r_num == 4:
            NumOfLabels = 44
        if r_num == 5 or r_num == 6:
            NumOfLabels = 66
        if r_num == 7 or r_num == 8:
            NumOfLabels = 88
        if r_num == 9 or r_num == 10:
            NumOfLabels = 110
        #print "NumOfLabels = %i" % NumOfLabels
        for i in range(NumOfLabels):
            labels.append(Label(self.verticalframe.interior, text=""))
            labels[i].pack()


        return_slots = clothes.get_return_list()
        positionx = [30,390,30,390,30,390,30,390,30,390]
        positiony = [25,25,425,425,825,825,1225,1225,1625,1625]

        for idx,val in enumerate(return_slots):
            imgin = Image.open(str(val)+".jpg")
            imgin = imgin.resize((300,400), Image.ANTIALIAS)
            cimage = ImageTk.PhotoImage(imgin)
            b = Radiobutton(self.verticalframe.interior, image = cimage, variable = return_index, value = val,
command = lambda:self.ReturnSelected())
            b.image = cimage
            b.place(x = positionx[idx], y = positiony[idx], height = 350, width = 300)
```

```
        #return_index.trace("w", self.ReturnSelected)

        HomeButton = Button(self.verticalframe.interior, text = "Home Page", command=lambda:
controller.show_frame(StartPage) )
        HomeButton.pack()



delete_index = IntVar()
class PageDelete(Frame):

    def DeleteSelected(*args):
        num = delete_index.get()
        if tkMessageBox.askyesno("Confirmation", "Are you sure to delete number %i clothes?" % (num +
1)):
            clothes.delete_clothes(num)
            if(num == 0):
                backwards(140)
            elif(num == 1):
                backwards(120)
            elif(num == 2):
                backwards(100)
            elif(num == 3):
                backwards(80)
            elif(num == 4):
                backwards(60)
            elif(num == 5):
                backwards(50)
            elif(num == 6):
                backwards(70)
            elif(num == 7):
                backwards(90)
            elif(num == 8):
                backwards(110)
            elif(num == 9):
                backwards(130)
            setStep(0,0,0,0)
            time.sleep(1)
            strip.setPixelColorRGB(num,0,255,255)
            strip.show()
            #purple light on(done)
            if tkMessageBox.askyesno("Confirmation", "Have picked?"):
                #purple light off(done)
                if(num == 0):
                    forward(140)
                elif(num == 1):
                    forward(120)
```

```
            elif(num == 2):
               forward(100)
            elif(num == 3):
               forward(80)
            elif(num == 4):
               forward(60)
            elif(num == 5):
               forward(50)
            elif(num == 6):
               forward(70)
            elif(num == 7):
               forward(90)
            elif(num == 8):
               forward(110)
            elif(num == 9):
               forward(130)
            setStep(0,0,0,0)
            time.sleep(1)
            strip.setPixelColorRGB(num, 0,0,0)
            strip.show()
          print "done"
        else:
            #after one minute light off(done)
            time.sleep(30)
            strip.setPixelColorRGB(num,0,0,0)
            strip.show()

def __init__(self, parent, controller):
    Frame.__init__(self, parent)
    self.verticalframe = VerticalScrolledFrame(self)
    self.verticalframe.pack()
    labels = []
    NumOfLabels = 0
    p_num = clothes.get_permanent_number()
    if p_num == 1 or p_num == 2:
        NumOfLabels = 22
    if p_num == 3 or p_num == 4:
        NumOfLabels = 44
    if p_num == 5 or p_num == 6:
        NumOfLabels = 66
    if p_num == 7 or p_num == 8:
        NumOfLabels = 88
    if p_num == 9 or p_num == 10:
        NumOfLabels = 110

    for i in range(NumOfLabels):
        labels.append(Label(self.verticalframe.interior, text=""))
        labels[i].pack()
```

```
        occupied_slots = [i for i, e in enumerate(clothes.permanent) if e != 0]
        positionx = [30,390,30,390,30,390,30,390,30,390]
        positiony = [25,25,425,425,825,825,1225,1225,1625,1625]

        for idx,val in enumerate(occupied_slots):
            imgin = Image.open(str(val)+".jpg")
            imgin = imgin.resize((300,400), Image.ANTIALIAS)
            cimage = ImageTk.PhotoImage(imgin)
            b = Radiobutton(self.verticalframe.interior, image = cimage, variable = delete_index, value =
val,command=lambda:self.DeleteSelected())
            b.image = cimage
            b.place(x = positionx[idx], y = positiony[idx], height = 350, width = 300)


        #delete_index.trace("w", self.DeleteSelected)

        HomeButton = Button(self.verticalframe.interior, text = "Home Page", command=lambda:
controller.show_frame(StartPage) )
        HomeButton.pack()


#num = clothes.get_permanent_number()
#print "Now there are %i clothes in the closet" % num
setStep(0,0,0,0)

try:

        clothes = Clothes()
        app = main(root)
        app.pack(side="top",fill="both",expand=True)
        root.mainloop()
except KeyboardInterrupt:
        setStep(0,0,0,0)
        GPIO.cleanup()
```