Abstract

This document contains the build instructions of our project divided into 3 parts Hardware, GUI, App and database.

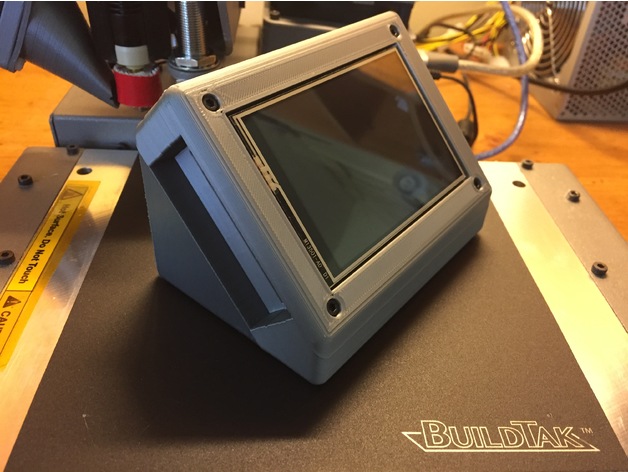
Store Helpline

Abhay Singla N01149096

Saqib Jaweed Syed N01152429

Guru Sharma N01142131

**Hardware Build Guide**

This page describes the build process for 3.5`` Raspberry Pi 3 Model B touch screen, RFID reader and PIR sensor but the process is the same for any model starting from Raspberry Pi 2.   
[](https://github.com/SaqibJaweed/Help-Line/blob/master/final_product.jpg?raw=true)

**Step 0: Read these Instructions**

Seriously. There are probably things in these later steps that you should know about before getting started. So read the complete build guide before you begin.

**Step 1: Order the Parts**

The good thing about this project is that you only need to order a few things that you can also order from any website even different from the ones mentioned below for a cheaper price.

1. One Raspberry Pi 3 Model B from <https://www.canakit.com/raspberry-pi-3-model-b.html?cid=cad&src=raspberrypi> for CA$45.95
2. One PiTFT - Assembled 480x320 3.5" TFT + Touchscreen for Raspberry Pi from <https://www.adafruit.com/product/2097>

QTY DISCOUNT  
1-9 CA$44.95  
10-99 CA$40.46  
100+ CA$35.96

1. One 16GB – 32GB class 10 Micro SD card from <https://www.amazon.com/SanDisk-Ultra-Micro-Adapter-SDSQUNC-016G-GN6MA/dp/B010Q57SEE/ref=sr_1_6?s=pc&ie=UTF8&qid=1516559163&sr=1-6&keywords=micro+sd> for CA$9.98
2. Universal power supply from <https://thepihut.com/collections/raspberry-pi-accessories/products/official-raspberry-pi-universal-power-supply> for CA$12 .00
3. One etchied PCB board with atleast 4 bus for $4.00 at Sayal Electronics.
4. One Long 40 pin Female - Male connector for $2.5 from Sayal Electronics
5. RFID Card Reader and Tags 1(reader) + 3(tags) for $59.99 from Parallax Inc
6. Female-Female Jumper Wires 10 $3.95 from Adafruit
7. One 2.2KOhm and One 3.3KOhm Resistor for $6.86 from Amazon.ca
8. One PIR Sensor for $8.99 from Amazon.ca

All the above materials will cost you Approx. CA$200.00 + tax

**Step 2: Time Commitment**

If you have decided to work on this project, make sure you give yourself enough time for all parts to arrive that can take upto 2 weeks to arrive unless you pay extra for fast delivery. But it does not ends here, even when the parts arrive, it will take atleast 6 to 8 hours of work to make the project from start to finish. Be careful to work in a neat and non-conductive environment, handling the touch screen and all other sensors with care and delicacy at all times.

**Step 3: Pre Install, SD Card Prep.**

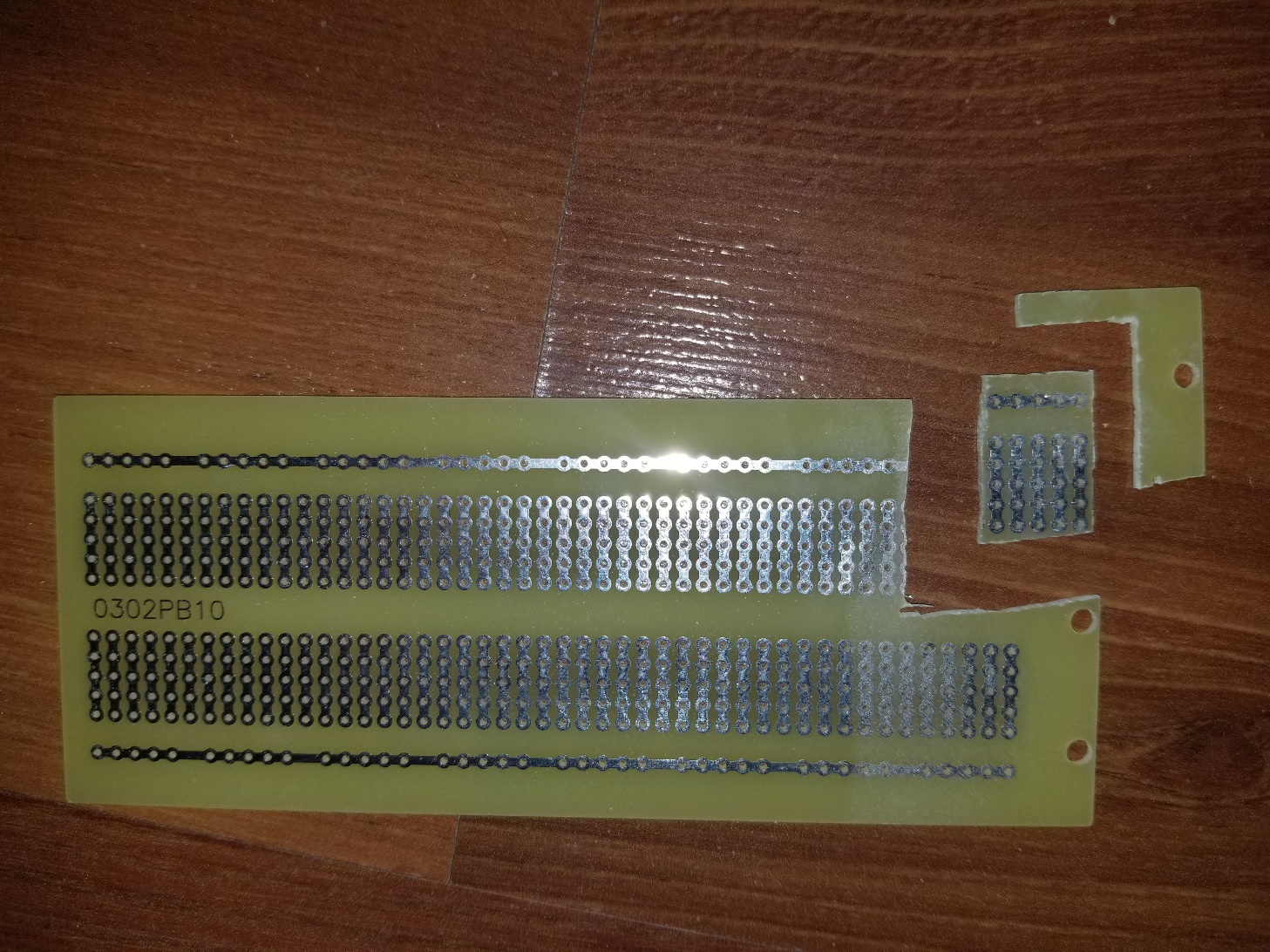
These instructions will prepare you for the later steps and contain instruction on how to install the Raspbian image on the SD card.

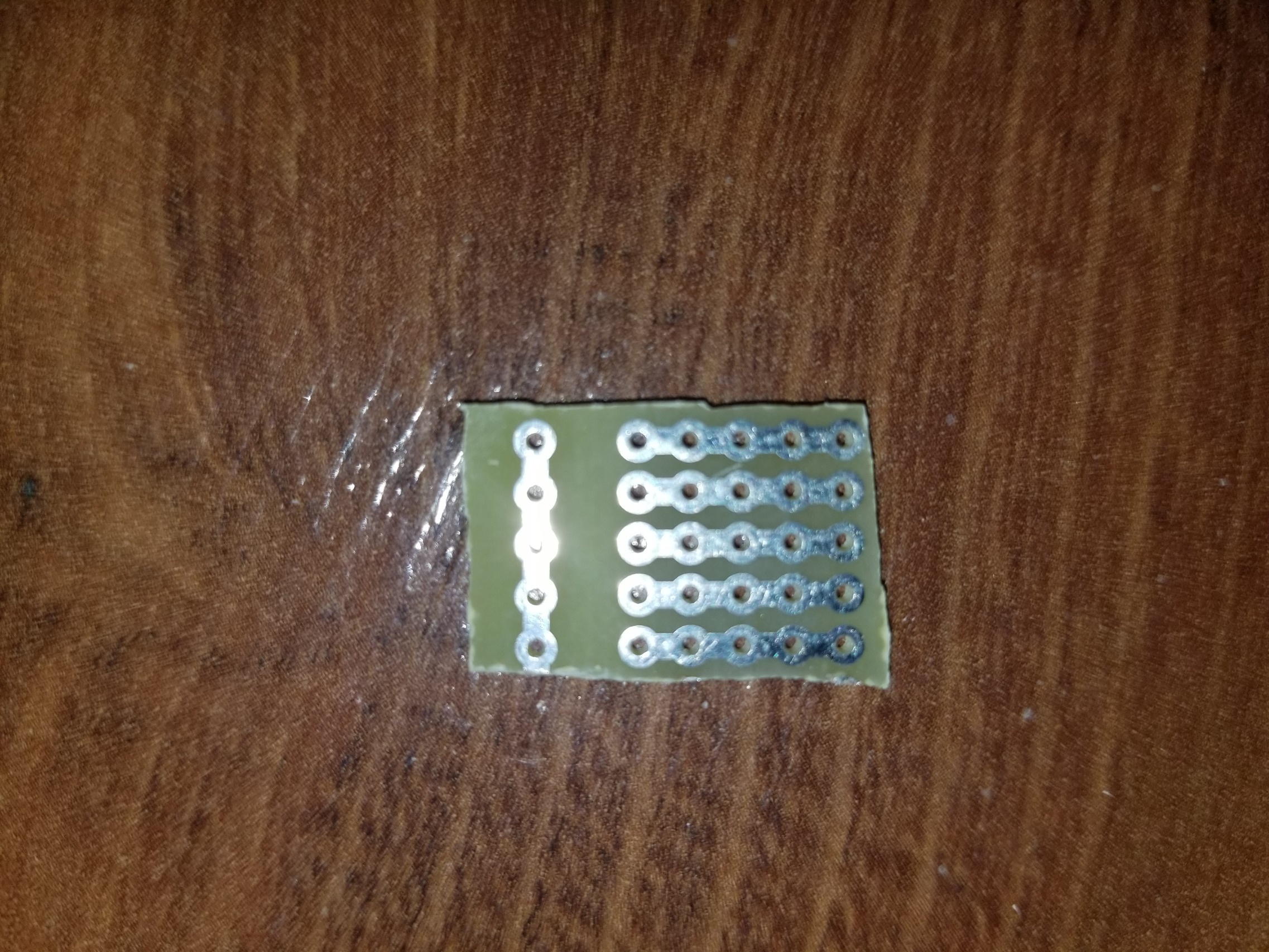
1. Insert your microSD card into your card reader and find out its drive letter in Windows Explorer (for example G:).
2. Format the card in FAT32 format.
3. Download Win32DiskImager from <https://sourceforge.net/projects/win32diskimager/> unzip the downloaded file and run the utility file.
4. Download the Latest Jessie image that contains the requires TFT display drivers from <https://drive.google.com/file/d/1LjGL4q_rgiJxwAZnhUNQZa28Kp7b6usB/view>
5. Select the Raspbian image file you downloaded.
6. Select the drive of your SD card in the ‘Device’ dropdown. Make sure you chose the correct one. Otherwise, you risk damaging the data on your hard drive.
7. Select ‘Write’ and wait for the process to finish which may take around 20 minutes to complete. That’s it!

**Step 4: Mechanical Assembly**  
The whole process of this project’s assembly is relatively easy if you follow the following insructions carefully:

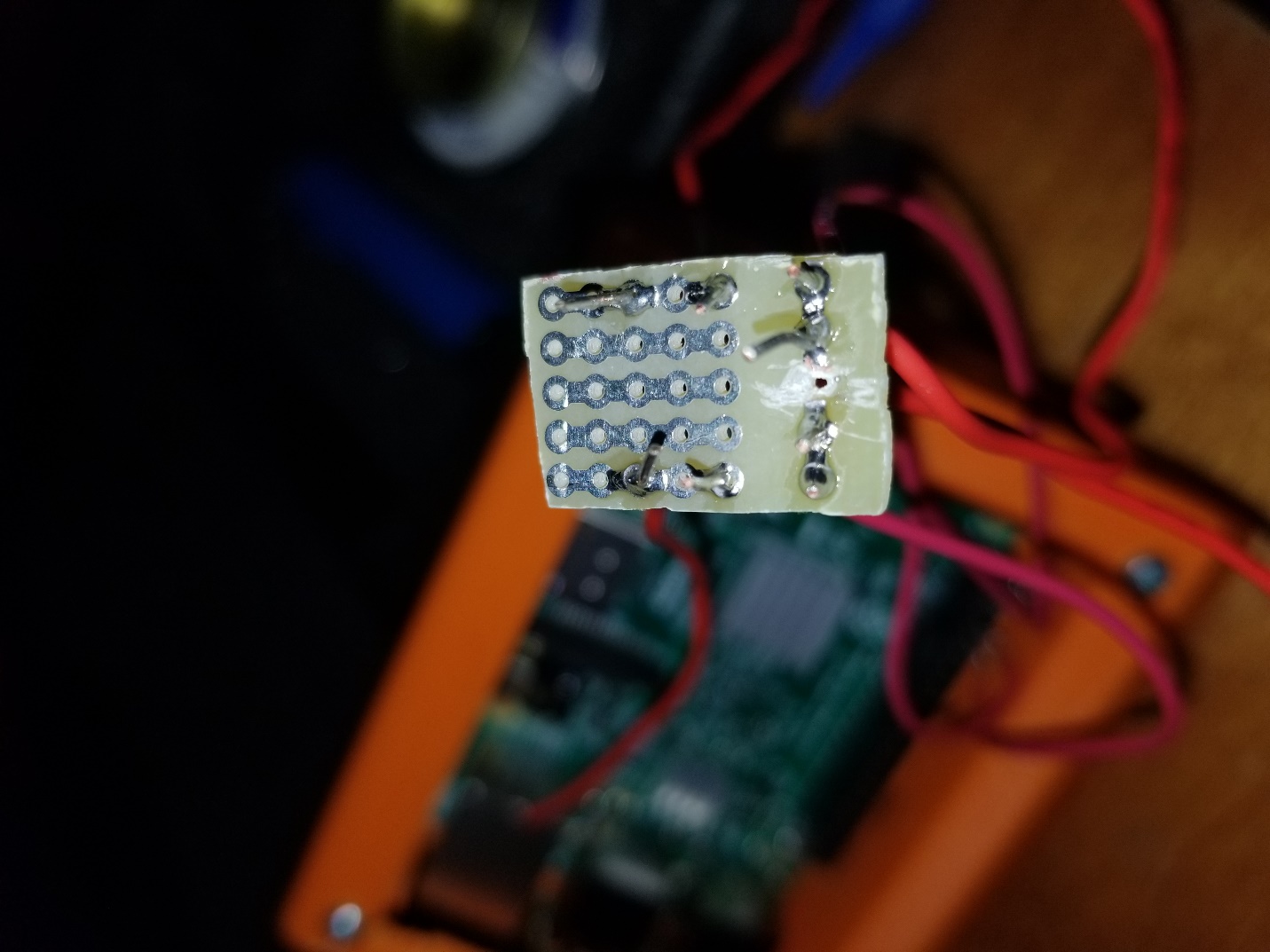
1. Place the Pi on a non-conductive surface with GPIO pins facing up
2. Solder jumper wires from the very bottom of pins 2, 4, 8 and 10 of a 40 pin connector and then place it firmly on top of the RPi’s pins.
3. Cut your Etiched PCB board to the following size using a hand saw or sharp utility knife carefully.



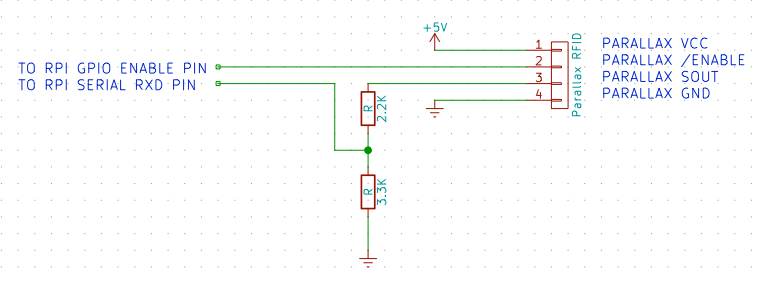




1. Solder the other side of each jumper wire to a separate bus of the PCB part you just cut out above.



1. Place the female pins of the screen onto the pins of the 40 pin connector, aligning them to the left most pins and then press firmly to make the pins go all the way in the adapter to make sure there are no loose parts.
2. Connect the RFID reader to its relative pins on the PCB using the following diagram



1. To connect the PIR Sensor, Use the 5V on the PCB, connect the output(middle) pin to GPIO12 @ pin 32 and finally use pin 34 for ground.
2. Insert the SD card in the SD card slot carefully with the specified Raspbian image in it.
3. Connect the power supply cable to the Raspberry Pi and let it sit for 2 minutes for the first time and the display will turn on and start recognizing inputs.
4. As the screen is small, it is recommended to use it with a stylus.

**Step 4.1: Extras**  
This part of the build instruction is only to be followed if you want to install your project in a 3D printed case:

1. Download the files from
2. Send the files for 3D printing that will come out in 3 parts and will need a minimum of 4 hours to complete
3. You will need 12 2.5mm Allen head bolts 6-10mm long
4. Break the mounting tabs from all 4 sides of the screen as follows and the install the screen on the Pi carefully and re-insert the SD card in the slot carefully.

[](https://github.com/SaqibJaweed/Help-Line/blob/master/tab_break.jpg?raw=true)

1. Arrange the pieces as shown and screw the RPi 3 on the case using four 2.5mm screws :  
   [](https://github.com/SaqibJaweed/Help-Line/blob/master/pi_install.jpg?raw=true)  
   [](https://github.com/SaqibJaweed/Help-Line/blob/master/screen_install.jpg?raw=true)
2. Arrange and Fix the parts together in the following order with 8 2.5mm Allen head bolts 6-10mm but do not tighten too much as it might snap the plastic



1. That is it! All ports and screen should now be usable with Raspberry Pi and Screen sitting on a 45 degree angle from our face, secured in a case.

**Unit Testing**

The Project looks absolutely stunning and makes it easy for the users to interact with the Pi using the touch screen. Since we printed the plug Friendly version of my case, the ports stick out of the case by 2mm so that it is easily compatible with every kind of input/output cable.

**Production Testing**

With the .img file available on hand, we can avoid the hassle of downloading the display drivers through the Raspberry Pi which takes 4 hours, fails often and needs an active internet connection throughout the process. Considering if we have the files and software downloaded, we only need 1 hour to reproduce the project. Imagine the speed in a case where you are producing 50 touchscreen computers in just over 50 hours.

**References:**

<https://www.thingiverse.com/thing:2471701>   
<https://learn.adafruit.com/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi/easy-install>

**GUI Documentation**

#BEFORE YOU START:

## Install these packages on the pi before we start the codes

## sudo pip install python-tk

## sudo pip install Pyrebase

#REFRENCES:

## https://pythonprogramming.net/tkinter-depth-tutorial-making-actual-program/

## https://gist.github.com/SirRobo/c503014bbb03088bec37a61231036461

## https://blog.devcolor.org/heating-up-with-firebase-tutorial-on-how-to-integrate-firebase-into-your-app-6ce97440175d

# The code for changing pages was derived from: http://stackoverflow.com/questions/7546050/switch-between-two-frames-in-tkinter

# License: http://creativecommons.org/licenses/by-sa/3.0/

import Tkinter as tk #The Tkinter library is used for python 2.7, the original code used tkinter which was for python 3.

import pyrebase #This library is an envelope for the python firebase library

config = { ## This config file is available on

"apiKey": "AIzaSyCHOCo1cKZs7nNrjo7Q2RSBWRpNzGQhSUk", ## the firebase console in the

"authDomain": "helpine-de9a7.firebaseapp.com", ## authentication tab under web setup

"databaseURL": "https://helpine-de9a7.firebaseio.com",

"projectId" : "helpine-de9a7",

"storageBucket": "helpine-de9a7.appspot.com",

"messagingSenderId" : "helpine-de9a7.appspot.com",

"serviceAccount": "/home/pi/Desktop/Sqube\_Solutions/helpine-70f2be3cc276.json" # This JOSN file will have to be downloaded from the service account

}

firebase = pyrebase.initialize\_app(config) # Initializing Fierbase on this app

H1\_FONT= ("Verdana", 14) ## Fonts to use in the GUI

H2\_FONT= ("Helvetica", 12)

class StoreHelpLine(tk.Tk): ## This class is the backbone of the app.

## It combines all the pages and also switches between the pages(classes)

def \_\_init\_\_(self, \*args, \*\*kwargs):

tk.Tk.\_\_init\_\_(self, \*args, \*\*kwargs)

container = tk.Frame(self)

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

container.grid\_rowconfigure(0, weight=1)

container.grid\_columnconfigure(0, weight=1)

self.frames = {} ## Container for all the pages

for F in (StartPage, Search\_Page, Call\_Help, Product\_Details): ## This loop fills the

## container with all the pages.

frame = F(container, self)

self.frames[F] = frame

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

self.show\_frame(StartPage)

def show\_frame(self, cont):

frame = self.frames[cont]

frame.tkraise()

# The Sign-In Screen

class StartPage(tk.Frame):

def Sign\_In(self): # Sign\_In button Event Handler

emailSt = emailEL.get() ## Email and password string for authentication

pwordSt = pwordEL.get()

auth = firebase.auth() # Initializing the auth()

user = auth.sign\_in\_with\_email\_and\_password(emailSt, pwordSt) # Passing email and password for authentication

if user is not None:

self.controller.show\_frame(Search\_Page) # If the authentication works then change the page

def \_\_init\_\_(self, parent, controller): ## The actual GUI is build in the \_\_init\_\_ method of each class

tk.Frame.\_\_init\_\_(self,parent)

global emailEL ## Made these variable global so that

global pwordEL ## the Sign\_In function can access them.

self.controller = controller

# Title

label = tk.Label(self, text="Store HelpLine", font=H1\_FONT)

label.grid(row=0,column=0,sticky="ew")

# Sub-Title

lable2 = tk.Label(self, text="Sign-In", font=H2\_FONT)

lable2.grid(sticky="ew")

# Email and Password labels

emailL = tk.Label(self, text='Email: ')

pwordL = tk.Label(self, text='Password: ')

emailL.grid(row=2, sticky="w")

pwordL.grid(row=3, sticky="w")

# Email and password text fields

emailEL = tk.Entry(self) # The entry input

pwordEL = tk.Entry(self, show='\*')

emailEL.grid(row=2, column=1)

pwordEL.grid(row=3, column=1)

# Sign In Button

button = tk.Button(self, text="Sign-In", command=self.Sign\_In)

button.grid(columnspan=2, sticky="w")

# Search Page for the item search

class Search\_Page(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

# Title

label = tk.Label(self, text="Store HelpLine", font=H1\_FONT)

label.grid(pady=10,padx=10)

# Search lable

searchl = tk.Label(self, text="Search", font=H2\_FONT)

searchl.grid(row=2, sticky="w")

# Search text box

searchEL = tk.Entry(self)

searchEL.grid(row=2, column=1)

# Buttons

button2 = tk.Button(self, text="Search",

command=lambda: controller.show\_frame(Product\_Details))

button2.grid(columnspan=2, sticky="w")

button1 = tk.Button(self, text="Sign Out",

command=lambda: controller.show\_frame(StartPage))

button1.grid(columnspan=2, sticky="w")

button3 = tk.Button(self, text="Request Help",

command=lambda: controller.show\_frame(Call\_Help))

button3.grid(columnspan=2, sticky="w")

class Product\_Details(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

label = tk.Label(self, text="Store HelLine", font=H1\_FONT)

label.grid(pady=10,padx=10)

label2 = tk.Label(self, text="PRODUCT SCREEN",font=H2\_FONT)

label2.grid(row=2)

button1 = tk.Button(self, text="Add to Cart")

button1.grid(columnspan=1, sticky="w")

button2 = tk.Button(self, text="Cancel",

command=lambda: controller.show\_frame(Search\_Page))

button2.grid(columnspan=1, column=2, row=3, sticky="w")

button3 = tk.Button(self, text="Request Help",

command=lambda: controller.show\_frame(Call\_Help))

button3.grid(columnspan=2, sticky="w")

class Call\_Help(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

label = tk.Label(self, text="StoreHelpLine", font=H1\_FONT)

label.pack(pady=10,padx=10)

label2 = tk.Label(self, text="Call For Help", font=H2\_FONT)

label2.pack(pady=10,padx=10)

text = tk.Label(self, text="Help is on its way!!")

text.pack(pady=10,padx=10)

button1 = tk.Button(self, text="OK",

command=lambda: controller.show\_frame(Search\_Page))

button1.pack(pady=10,padx=10)

app = StoreHelpLine()

app.mainloop()

**Application Build Guide**

**Introduction:**

The purpose of this app is to allows users to see their shopping cart. The app is linked with real time Firebase database, where users and their cart information is stored. After logging into the app, the users will be able to see their respective shopping carts. The app will allow users to see the description of an item by clicking on it in the list view. Users can also remove items from the cart by using the app and all the changes made to the cart will update the database within seconds.

**App classes:**

The app has four main classes, namely:

1. MainActivity
2. SignUp
3. ShoppingCart
4. ProductDesription

**MainActivity:** It is the first screen that is launched when the user starts the app. It gives the user an option to either Sign In or Sign Up with their email and password.

1. 

New users will have to Sign Up first in order to log in to the app. Input fields on this page have validation on them in order to make sure that the user enters the data in the correct format. After successful sign in, the user will be directed to the shopping cart page.

**SignUp:** The purpose of this class is to allow new users to Sign Up with a valid email and password and after successful attempt, it redirects them to the MainActivity class where user can Sign In to the app after using the credentials they created.

1. 

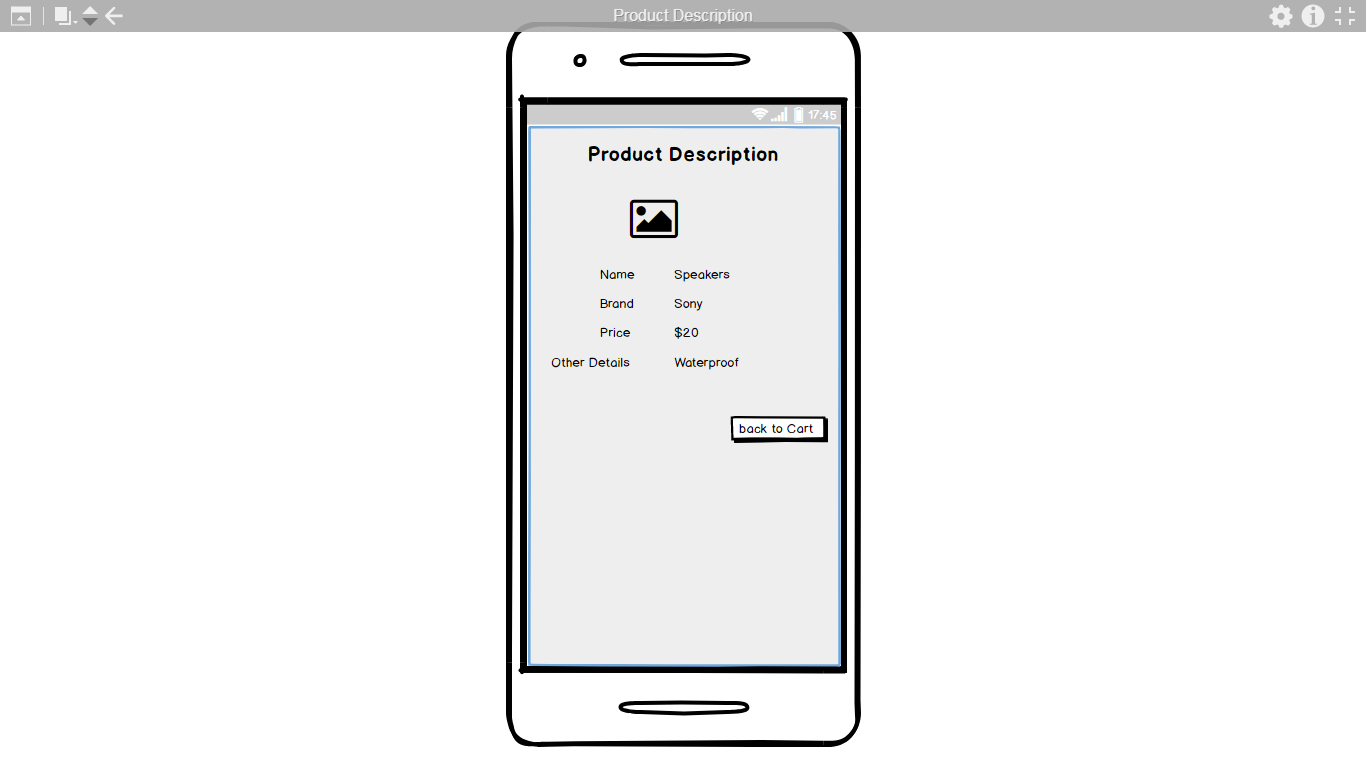
Sign Up page will allow first time users to Sign Up. After successfully signing up, the users will be redirected to the MainActivity page, where they have to Sign In to be able to see their shopping cart.

**ShoppingCart:** This class allows users to see the items in their shopping cart.

1. 

This is the page where users can see their shopping cart. This page is linked with real time database on the Firebase. Users can add items to the shopping cart from the interactive touch screen (part of hardware component), in order to see them in the app. Since our interactive touch screen will also be linked with the Firebase, any changes made to the shopping cart there, will be reflected immediately on the Android app. From within the app, the users will have the option to remove an item from their cart and they can also see the product details by clicking on the item. On this page the users will also have the option to log out from the app and they will be able to create a new account or sign in from a different account in the app.

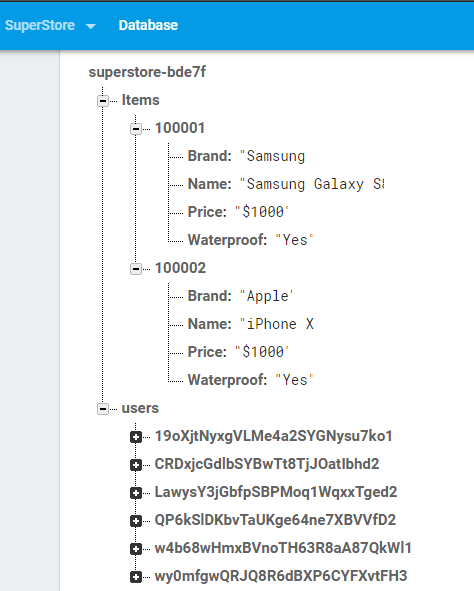
**ProductDesription:** This class allows users to see the product description.

1. 

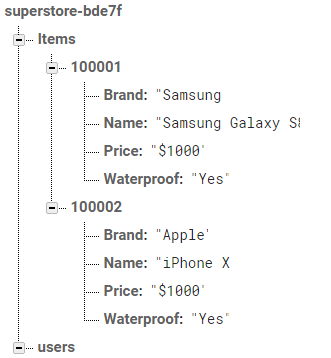
The users will be directed to this page after they click on an item in the list view of their cart (from ShoopingCart page) with an intention to see the details of the product. The page will have some details (including name and price) and a picture of the item. The user can return to cart by clicking on the "back to cart" button.

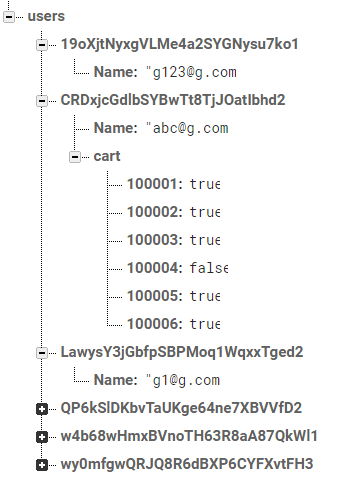
**Database Build Instructions**

We are using Google's Firebase real-time database as the database for our app, because the shopping cart needs to reflect changes within a short period of time. Below is the structure of the database that we are using. Information for the items is stored under "Items" node and the details of users who successfully sign up are listed under "users" node and are represented by a unique ID (with numbers and letters).



We have created unique SKUs (stock keeping units) for our items in the database. When the user adds an item to the shopping cart, this is the unique ID that will be used to store an item in the user's cart. This unique ID is then utilized to refer back to this particular item to retrieve and show its details in the product description.





When the user signs up a new node gets created under "users" node with a unique ID generated by the Firebase. When a user logs into the app this unique ID is used to show the shopping cart that belongs to the signed in user.