دانشکده مهندسی

گروه کامپیوتر

پایان‌نامه کارشناسی

گرایش نرم‌افزار

عنوان: دربازکن تصویری هوشمند

دانشجویان: گروه ۱) امیرحسین نجفی، مجتبی هاشمی پر چینی

گروه ۲) پارسا کاظمی، علی اسلامی

استاد راهنما: دکتر علی امیری

(تاریخ دفاع: تیر ۱۴۰۱)

نیم سال دوم ۰۱-۱۴۰۰


بسم الله الرحمن الرحیم

فرم نمره نهایی پروژه کارشناسی- گروه کامپیوتر دانشگاه زنجان

نام و نام خانوادگی دانشجو	امیرحسین نجفی، مجتبی هاشمی پرچینی
شماره دانشجویی	۹٦٤٦۳۱٥٦ ، ۹٦٤٦۳۱٦۳
عنوان پروژه	دربازکن تصویری هوشمند
نمره نهایی به عدد	
نمره نهایی به حروف	
تاریخ دفاع	تیر ۱٤۰۱
نام و امضای استاد راهنما	
نام و امضای مدیر گروه	

فرم نمره نهایی پروژه کارشناسی- گروه کامپیوتر دانشگاه زنجان

نام و نام خانوادگی دانشجو	علی اسلامی، پارسا کاظمی
شماره دانشجویی	۹٦٤٦۳۱۰٥ ، ۹٦٤٦۳۱٤۱
عنوان پروژه	دربازکن تصویری هوشمند
نمره نهایی به عدد	
نمره نهایی به حروف	
تاریخ دفاع	تیر ۱٤۰۱
نام و امضای استاد راهنما	
نام و امضای مدیر گروه	

فهرست مطالب

عنوان صفحه

فهرست اشکال و جداول

فصل دوم

فصل سوم

فصل چهارم

فصل پنجم

فصل ششم

چکیده

دربازکن‌های رایج امکان تشخیص فرد را ندارند و اگر صاحب‌خانه دچار اشتباه تشخیص بشود، ممکن است در را برای دیگران باز بکند یا در مواردی ممکن است کلید فراموش‌شده باشد و کسی در خانه نباشد تا در را باز کند یا کسی که در خانه است متوجه صدای زنگ نشود و صدها نوع اتفاق که ممکن است مشکل ایجاد کند.

در این پروژه سعی شده است تا دستگاهی برای حل این مشکلات ساخته شود.

فصل اول

مقدمه

۱-۱) شرح مسئله

دربازکن‌های رایج مشکلات زیادی به همراه دارند و در طی زمان با پیشرفت‌های فراوانی مثل دوربین، صدا و غیره، ولی همچنان مشکلاتی رادارند مثل‌اینکه حتماً شخصی باید به دربازکن ساختمان دسترسی داشته باشد و در غیر این صورت باید با کلید یا اثرانگشت و مشابه آن باز بشود.

۱-۲) هدف از طراحی

دربازکن هوشمند با پردازش تصویر این امکان را می‌دهد تا اعضایی که از قبل تعریف‌شده هستند بدون دخالت شخص سومی وارد خانه بشوند و در را برای آن‌ها باز می‌کند، صرف‌نظر از اینکه چه تغییراتی در چهره و لباس آن‌ها ایجادشده است. صاحب‌خانه هم می‌تواند با در دست داشتن اپلیکیشن مدیریتی دستگاه، روی تاریخچه عبور و مرور، لیست اعضایی که اجازه ورود دارند و حتی تشخیص کسانی که اجازه ورود نداشته‌اند ولی واردشده‌اند، نظارت داشته باشد. البته این دستگاه روند دربازکن‌های قبلی را هم با خود دارد و حتی صاحب دستگاه می‌تواند با اپلیکیشن در را باز کند.

روند ساخت دستگاه به دو بخش نرم افزار و سخت افزار تقسیم میشود که در ادامه به بررسی آن‌ها می‌پردازیم.

بخش اول

نرم افزار

آنچه در بخش نرم افزار مطالعه خواهید کرد :

اولین قدم در ساخت دربازکن هوشمند ، پیاده سازی مجموعه نرم افزاری بود تا دستگاه بتواند با آن روند تعریف شده را انجام دهد.

مجموعه نرم افزاری که مورد نیاز ساخت این دستگاه میباشد در لیست یزر آورده شده است:

– **پایگاه داده** : تمامی اطلاعات کاربران و دستگاه نیاز به ذخیره سازی در یگ پایگاه داده قابل اطمینان دارد ؛ همه روند های اجرایی توسط دستگاه باید در این مکان ذخیره بشوند تا در صورت نیاز ، به کاربر بتواند روی آنها نظارت داتشته باشند ، مثل اعضایی که اجازه ورود دارند ، تاریخچه عبور و مرور و ... .

– **سرور** : هر برنامه ایی که میخواهد با کاربر ارتباط برقرار کند ، نیاز به یک سرور دارند تا ارتباط بین این دو را برقرار کرده و روی آن نظارت داشته باشد.این روند شامل ارتباط هایی مثل :ارسال دستور باز شدن در ورودی به کاربر ، ارسال تغییراتی که در سمت کاربر یا دستگاه اتفاق می افتد و .. میباشد.

– **اپلیکیشن تلفن همراه** :کاربر باید از طریق برنامه ایی با دستگاه ارتباط برقرار کند .در دربازکن های مرسوم ، یک پنل مدیریتی بر روی آنها پیاده سازی شده تا دستورات باز کردن یا دیدن از طریق دوربین را به دستگاه بدهد. در دربازکن هوشمند ،به دلیل اینکه روند های اضافه کردن عضو ، دیدن تاریخچه عبور و مرور ، حذف عضو و ... وجود دارد ، اپلیکیشنی پیاده سازی شده است تا کاربر با استفاده از آن ، روی دستگاه نظارت داشته باشد.

– **پردازش تصویر** : برنامه دیگری که دستگاه در پشت پرده اجرا میکند ، روندی از الگوریتم های هوش مصنوعی میباشد که با آن ، چهره عضو هایی که تعریف شده اند و اجازه دسترسی دارند را تشخیص میدهد و همچنین قابلیت تشخیص چهره واقعی و غیر واقعی (مثل عکس) هم دارد.این روند با پردازش تصویر انجام میشود و درنهایت ، دستگاه براساس آن ، به سرور اعلام میکند که درباز بشود یا نشود.

فصل دوم

پایگاه داده

۲-۱) نیازمندی‌های داده‌ای

دربازکن هوشمند از اطلاعات عضوهایی که اجازه ورود دارند یا ندارد استفاده زیادی می‌کند و به دلیل اینکه صاحب‌خانه می‌تواند بر روی تاریخچه‌ی ورود نظارت داشته باشد، به پایگاه داده ایی نیاز است که تمامی اطلاعات در آن ذخیره بشود.

۲-۲) فناوری‌های پایگاه داده

برای مدیریت داده‌ها و ذخیره‌سازی آن از ORM موجود در جنگو و دیتابیس Postgresاستفاده شد (دلیل استفاده از این زبان در فصل بعد موردبررسی قرار می‌گیرد) که هرکدام از برنامه‌ها شامل مدل‌های مختلف موردنیاز خود است که در ادامه به آن‌ها می‌پردازیم.

به این دلیل از ORM استفاده‌شده است که باعث می‌شود تا حدودی از درگیری با بحث‌های sql شامل ایجاد، درج، حذف، آپدیت جلوگیری کرد، به‌طورکلی ما را از انجام QuerySet راحت می‌کند، از طرفی مدیریت پایگاه داده را به خود جنگو که شامل اتصال به پایگاه داده، پیش آمد خطا، قطعی و...واگذار می‌کند.

۲-۳) طراحی پایگاه داده

هر یک از مدل‌هایی که توسط ORM ایجاد و در پایگاه داده به‌صورت جدول پیاده‌سازی می‌شود که ابتدا دستور python manage makemigrations را در کنسول اجرا می‌کنیم تا migration های مربوط به هر مدل ایجاد شود که شامل کویری‌های مدل‌ها برای ایجاد جدول می‌باشد. سپس دستور python manage migrate برای migration های ایجادشده در پایگاه داده، اجرا و جداول ایجاد می‌شوند.

برای پایگاه داده از Postgres استفاده‌شده است که یک پایگاه داده RDBMS به معنی سیستم مدیریت پایگاه داده رابطه‌ای یک DBMS می‌باشد که به‌طور خاص برای پایگاه داده‌های رابطه‌ای طراحی‌شده است. از سمت دیگر این پایگاه داده به‌صورت رایگان از طریق heroku قابل تهیه و ایجاد کردن می‌باشد. امروزه هم بسیاری از برنامه‌نویسان از این پایگاه داده استفاده می‌کنند و سهم عمده‌ای از پایگاه داده‌های رابطه‌ای را به خود اختصاص داده است.

پس ایجاد یک app درسایت heroku می‌توان postgres را ایجاد و اطلاعات مربوط به اتصال به آن را دریافت کرد. برای ذخیره عکس‌ها ابتدا عکس به فرمت base64 که یک نوع داده از نوع رشته می‌باشد تبدیل و در یک فیلد از نوع متن (text) ذخیره می‌شود.

حال به بخش مدل‌های ایجادشده توسط ORM جنگو در هر یک از app های پروژه هست می‌پردازیم.

۲-۳-۱) حساب کاربران

مدل حساب کاربران (Profiles): این مدل برای مدیریت کاربران ایجادشده که با ایجاد ارتباط با user جنگو بحث login و... توسط جنگو هندل می‌شود بقیه فیلدهایی که برای کاربر نیاز است ولی در مدل user جنگو وجود ندارد به این مدل اضافه می‌شود.

Column Name	#	Data type	Length	Precision	Scale	Identity	Collation	Not Null	Default	Comment
id	1	int8		19			_	☑	nextval('accounts_profile...	
mobile	2	varchar	11	11			_	☑		
user_id	4	int4		10			_	☑		
serial_reset_password	5	varchar	12	12			_	☑		

شکل ۲-۱) جدول حساب کاربری

مدل پیامک احراز هویت (AuthSMS): این مدل برای ذخیره اطلاعات پیامک‌های ورود، تغییر رمز و...است که شامل کدتانید، تاریخ، زمان ارسال،hash ارسال‌شده برای کاربر برای ارسال دوباره و تائید کدتانید، نوع پیامک، وضعیت پیامک، فیلد کلید خارجی به Profiles هست.

Column Name	#	Data type	Length	Precision	Scale	Iden...	Collation	Not Null	Default	Co
id	1	int8		19			...	☑	nextval('accounts_authsms_id_seq'::regcl...	
timeSend	2	timestamptz		35	6		...	☑		
codeSended	3	int4		10			...	☑		
token	4	varchar	300	300			...	☑		
type_SMS	5	int4		10			...	☑		
state_SMS	6	int4		10			...	☑		
profileUser_id	7	int8		19			...	☑		

شکل ۲-۲) جدول احراز هویت

۲-۳-۲) امنیت درب ورودی

مدل اطلاعات سرویس (InformationService): این مدل شامل اطلاعات سرویس در می‌باشد. شامل وضعیت قابلیت باز کردن درب، ارتباط به مدل اطلاعات رزپری‌پای در app مربوطه، عنوان سرویس ارائه‌شده، ارتباط به مدل لاینسس می‌باشد.

Column Name	#	Data type	Length	Precision	Scale	Iden...	Collation	Not Null	Default	Co
id	1	int8		19			...	☑	nextval('"doorSecurity_informationservic...	
status_opendoor	2	int4		10			...	☑		
lincense_id	4	int8		19			...	☑		
rassperypilnfo_id	5	int8		19			...	☑		
title	6	varchar	30	30			...	☑		

شکل ۲-۳) جدول اطلاعات دستگاه

مجوز دسترسی (LicenseToUse): این مدل شامل اطلاعات لاینسس برای دسترسی کاربر به سرویس‌های می‌باشد. شامل تاریخ شروع و پایان لاینسس می‌باشد.

Column Name	#	Data type	Length	Precision	Scale	Iden...	Collation	Not Null	Default	Co
start_lincense	1	date		13				☑		
id	2	int8		19				☑	nextval('"doorSecurity_licensetouse_id_s...	
end_lincense	3	date		13				☑		

شکل ۲-٤) جدول مجوز دسترسی

تاریخچه (history): این مدل شامل اطلاعات افرادی است که روبروی دستگاه قرارگرفته‌اند تا درباز شود که شامل تاریخ، وضعیت درخواست، عکس، فیلد ارتباط با مدل اطلاعات رزری‌پای، فیلد ارتباط با اعضا (درصورتی‌که فرد توسط دستگاه شناخته شد پر می‌شود.) هست.

Column Name	#	Data type	Length	Precision	Scale	Iden...	Collation	Not Null	Default	Co
id	1	int8		19			...	☑	nextval('"doorSecurity_history_id_seq"'::r...	
date	2	timestamptz		35	6		...	☑		
request_status	3	int4		10			...	☑		
member_id	4	int8		19			...	☐		
rassperypilnfo_id	5	int8		19			...	☑		
picture	6	text					...	☑		

شکل ۲-۵) جدول تاریخچه

مدل اعضا (member): این مدل شامل اطلاعات افرادی است که کاربر برای شناخته شدن آن‌ها توسط دستگاه اضافه‌شده است. شامل تاریخ اضافه شدن، عنوان، نام، وضعیت اجازه ورود، عکس، تاریخ آخرین تغییر وضعیت، فیلد ارتباط با سیستم رزپری‌پای هست.

Column Name	#	Data type	Length	Precision	Scale	Iden...	Collation	Not Null	Default	Co
add_date	1	timestamptz		35	6		...	☑		
id	2	int8		19			...	☑	nextval('"doorSecurity_members_id_seq"...	
name	3	varchar	20	20			...	☑		
change_status_date	4	timestamptz		35	6		...	☑		
allow_status	5	int4		10			...	☑		
rassperySystem_id	6	int8		19			...	☑		
title	7	varchar	20	20			...	☑		
picture	8	text					...	☑		

شکل ۲-۶) جدول اطلاعات اعضا

۲-۳-۳) اطلاعات رزبری پای

مدل سیستم رزپری‌پای (RassperySystem): شامل اطلاعات رزپری‌پای نصب‌شده در دستگاه و به‌طورکلی اطلاعات کلی از دستگاه و محل نصب. شامل سریال رزپری‌پای در دستگاه،hash سریال رزپری‌پای، توکن اتصال به سرور توسط رزپری‌پای، فیلد ارتباط با پروفایل صاحب دستگاه، وضعیت دستگاه (آنلاین یا آفلاین بودن)، نوع رزپری‌پای به‌کاررفته، آدرس، عنوان می‌باشد.

Column Name	#	Data type	Length	Precision	Scale	Iden...	Collation	Not Null	Default	Co
id	1	int8		19			...	☑	nextval('"rassperypilnfo_rasspersystem_...	
token_connect_rassperypi	2	varchar		256			...	☑		
serial_rasperyPi	3	varchar	16	16			...	☑		
profile_id	5	int8		19			...	☑		
online_status	6	int4		10			...	☑		
address	7	text					...	☑		
hash_serial_rassperyPi	8	varchar	64	64			...	☐		
title	9	varchar	80	80			...	☑		
type_rassperyPi	10	int4		10			...	☑		

شکل ۲-۷) جدول اطلاعات دستگاه رزبری پای

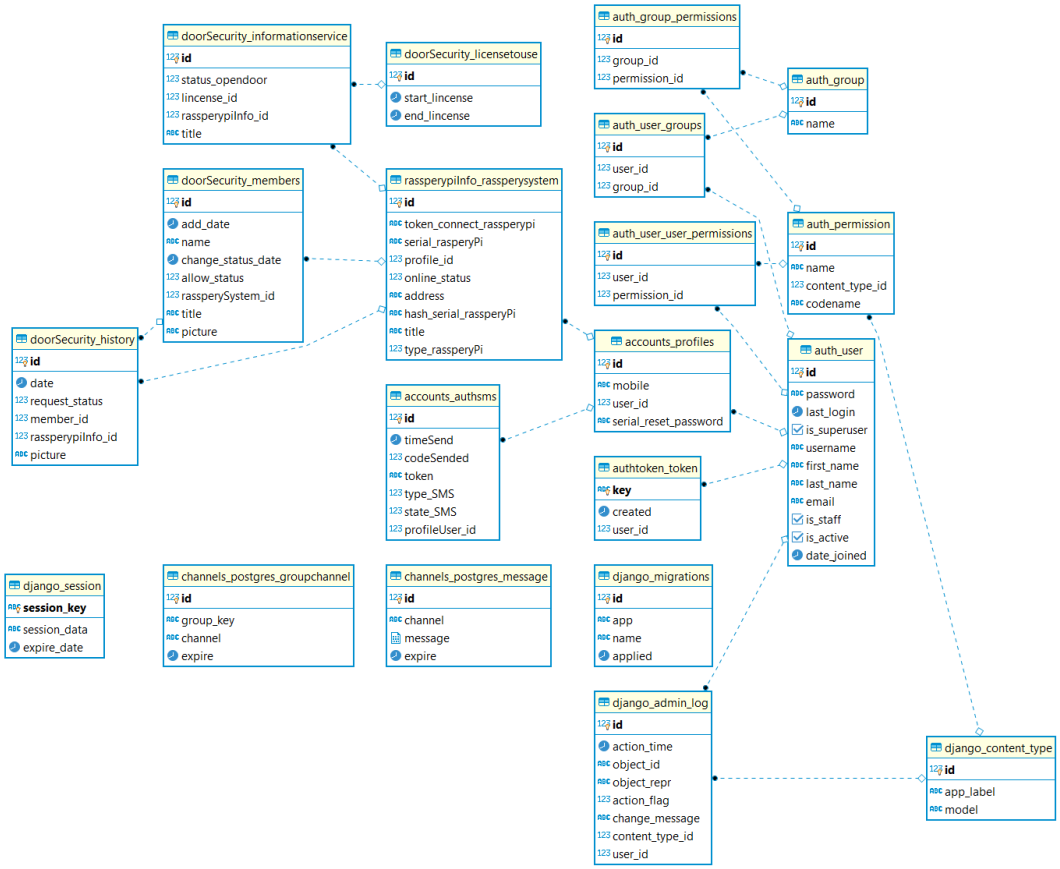
از طرفی برای اینکه سریال رزپری‌پای به‌طور خودکار هنگام اضافه کردن سیستم جدید hash و ذخیره شود از تابع save() در کلاس مدل به‌صورت زیر استفاده‌شده است:

```
def save(self, *args, **kwargs):
    self.hash_serial_rassperyPi = hashlib.sha256(self.serial_rasperyPi.__str__().encode('utf-8')).hexdigest()
    # Call the original save method
    super(RassperySystem, self).save(*args, **kwargs)
```

شکل ۲-۸) تابع ذخیره‌سازی سریال دستگاه

کاربرد تابع به این صورت است که به هنگام ایجاد یک رکورد جدید و در هنگام ذخیره آن اجراشده و فیلد hash سریال رزپری پای مقدار آن را ایجاد کرده و قرار می‌دهد.

به‌طورکلی شمای پایگاه داده ذکرشده به‌صورت زیر می‌باشد:



شکل ۲-۹) شمای روابط جدول‌های پایگاه داده

فصل سوم

سرور

همان‌طور که در فصل قبل مطرح شد ، برای طراحی پایگاه داده و سرور برنامه ، از Django استفاده شد. با این زبان ، app های مختلفی را می‌توان پیاده‌سازی کرد که درنهایت به هدف سرور ما ؛ که ارائه چندین سرویس برای دستگاه است ،کمک می‌کند.

۳-۱) app های به‌کاررفته در برنامه:

Back end پروژه داری پنج app زیر است:

accounts: شامل مدیریت کاربران برنامه هست که برای این منظور از user جنگو که قبلاً گفته شد استفاده می‌شود.

doorSecurity: بخش درب پروژه بوده.

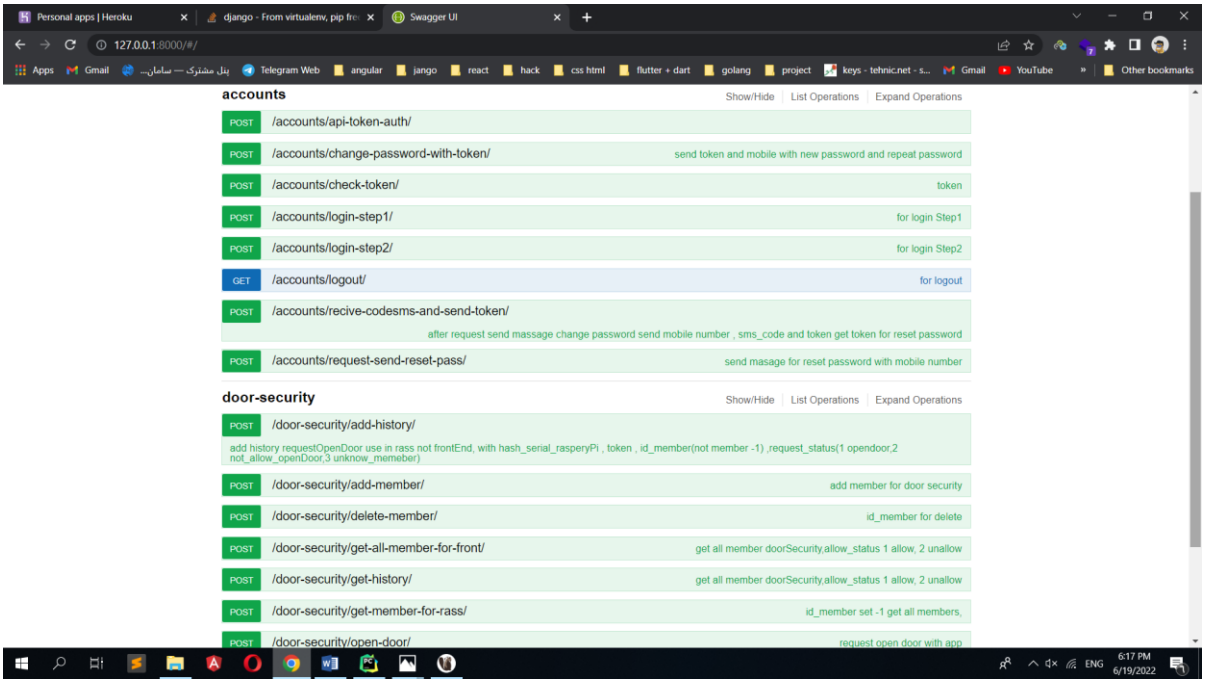
generallApp: قسمت از پروژه می‌باشد که در شامل چندین app دیگر است.

rassperypiInfo: مدیریت رز پری‌های موجود در شبکه می‌باشد.

websocketManage : وظیفه ایجاد ارتباط به‌صورت websocket با دستگاه است که قرار است با سرور ارتباط برقرار کنند.

۳-۲) مفاهیم به‌کاررفته در طراحی :

برای ارتباط با سرور در پروژه از Django-rest-framework که یک RESTFUlL می‌باشد استفاده‌شده است.



شکل ۳-۱) شمای api ها در swagger

از websocket برای ارتباط دستگاه با سرور استفاده‌شده است که برای مدیریت سوکت‌ها و ارسال پیام نیاز به دیتابیس دارد برای این منظور از همان دیتابیس اصلی پروژه(postgres) استفاده‌شده است.همان‌طور که در دیاگرام پایگاه داده در بخش قبل دیده شد دو جدول زیر : channels_postgres_groupChannel و channels_postgres_message برای این منظور بعد از اضافه کردن تنظیمات websocket در فایل settings سرور و اجرا دو دستور makemigrations و migrate که در بخش پایگاه داده گفته شد به‌صورت خودکار ایجاد می‌گردد.تا دستگاه به در صورت آنلاین بودن و ارتباط با سرور پیام‌ها،کنترلی را به‌صورت REAL_TIME دریافت کند این پیام‌ها به‌صورت زیر می‌باشند:

```
1011 for request requestOpenDoor
1012 for on detect and requestOpenDoor
```

```
1013 for off detect and requestOpenDoor
1014 for add member
1015 for update members
1016 for ok
1017 for  not exit member
1018 for delete member
```

شکل ۳-۲)کنسول کنترلی دستگاه

که هر کد به هنگام اتفاق افتادن کار مربوطه خود به همراه اطلاعات موردنیاز آن اتفاق برای دستگاه ارسال می‌گردد تا از رویداد اتفاق افتاده خود باخبر شود و کارهای مربوطه با آن اتفاق را انجام دهد.

به‌طور نمونه یک پیام که از سمت سرور برای دستگاه توست websocket ارسال می‌شود به‌صورت زیر می‌باشد:

```
{'massege': 'update member', 'code': 1015, 'id_member': member.id}
```

شکل ۳-۳)نمونه پیام سیستم (تغییر اطلاعات عضو)

این پیام مربوط به این است که کاربر یک عضو اطلاعاتش تغییر کرده است برای باز کردن در اضافه کرده است و دارای id عضو می‌باشد وقتی دستگاه این پیام را دریافت کرد از طریق api که برای این کار در نظر گرفته‌شده است اطلاعات کاربر را یک‌بار دیگر دریافت کرده و بروز رسانی می‌کند.

۳-۳) پیاده‌سازی Api :

API ها به‌طورکلی به دو بخش تقسیم می‌شوند: .apiهای اپلیکیشن .api های دستگاه

apiهای اپلیکیشن:

۳-۳-۱) Accounts: Urlهای مربوط به این app شامل زیر می‌باشد:

```
urlpatterns=[
    path('login-step1/', api.loginStep1Api.as_view(), name='ورودکاربر مرحله اول'),
    path('login-step2/',csrf_exempt(api.loginStep2Api.as_view()),name='ورود کاربر مرحله دو'),
    path('logout/', csrf_exempt(api.LogoutApi.as_view()), name='خروج کاربر'),
    path('request-send-reset-pass/', api.SendMassegeToResetPasswordAndGetTokenApi.as_view(), name='ارسال پیام احرازهویت ریست پسوورد و دادن توکن'),
    path('recive-codesms-and-send-token/', api.ReciveCodeSmsTokenAndSendTokenApi.as_view(), name='دریافت توکن و ارسال توکن SMS'),
    path('change-password-with-token/', api.ChangePasswordWithTokenApi.as_view(), name='دریافت توکن و تغییر رمز'),
    path('api-token-auth/', views.obtain_auth_token),
    path('check-token/', api.checkToken.as_view(),name="برای چک کردن توکن"),
]
```

شکل ۳-٤) url های مربوط به‌حساب کاربری

برای ورود کاربر به اپلیکیشن از احرازهویت دومرحله‌ای بر پایه پیام کوتاه استفاده‌شده است که دو api ، login-step1/ و login-step2/ برای این منظور است.

برای ارسال پیام کوتاه از سامانه SMS.ir استفاده می‌شود که به‌صورت رندوم یک کد ٦ رقمی ایجاد می‌شود و با api زیر برای سامانه ارسال تا آن را برای کاربر پیامک کند:

```
try:
    url = 'https://restfulsms.com/api/MessageSend'
    headers = {'Content-Type': 'application/json', 'x-sms-ir-secure-token': token}
    body = {"Messages": [str(text_sending) + str(code_sending)], "MobileNumbers": mobile,
            "LineNumber": "30004603370615", "SendDateTime": "", "CanContinueInCaseOfError": "false"}
    response = requests.post(url, headers=headers, json=body)
except:
    return False
if response.json()['IsSuccessful'] :
    return True
elif flag ==0:
    status_update_token=getTokenSmsSend()
    if status_update_token:
        sendSmS(text_sending,code_sending,mobile,1)
    else:
        return False
else:
    return False
```

شکل ۳-٥)Api استفاده‌شده در سامانه پیامکی

در login-step1 اپلیکیشن شماره موبایل و نام کاربری که کاربر برای ورود وارد کرده است را ارسال می‌کند

از آن‌طرف سرور در صورت که serializers، api به‌صورت زیر است:

```
class LoginStep1Serializer(serializers.Serializer):
    mobile = serializers.CharField(max_length=11,min_length=11)
    password = serializers.RegexField(regex=r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$')
```

شکل ۳-۶)کد Api استفاده‌شده در login

اطلاعات به‌صورت صحیح پرشده بود ادامه api اجرا می‌شود در غیر این صورت پیام ۴۰۰ برگردانده می‌شود.حال بررسی می‌شود شماره تلفن و پسوورد باهم تطابق دارند و برای کاربر می‌باشد در این صورت یک پیامک حاوی کدتانید برای کاربر ارسال می‌شود سپس در جواب اپلیکیشن یک پیام تائید حاوی یک توکن برای مرحله بعد ارسال می‌شود:

```
class loginStep1Api(APIView):
    schema =schemas.loginStep1Schema()
    def post(self, request, *args, **kwargs):
        serializer = serializers.LoginStep1Serializer(data=request.data)
        if serializer.is_valid():
            data = serializer.validated_data
            mobile = data.get('mobile')
            password = data.get('password')

            try:
                profile = Profiles.objects.get(mobile=mobile)
                user = authenticate(request,
username=profile.user.username, password=password)
                if user is  None:
                    return Response({"message": "Username or Password is
incorrect."},
                                    status=status.HTTP_401_UNAUTHORIZED)

            except:
                return Response({"message": "Username or Password is
incorrect."}, status=status.HTTP_401_UNAUTHORIZED)
            sms_code = random.randint(100000, 999999)
            status_send_sms = smsHandeller.sendSmS(text_sending="ارسالی کد
احرازهویت:", code_sending=sms_code,
                                                   mobile=[mobile],
flag=0)
            sms_code=123456
            status_send_sms=True
            if status_send_sms :
                for authSms in
AuthSMS.objects.filter(profileUser=profile, state_SMS=1,type_SMS=2):
                    authSms.state_SMS = 2
```

```
                    authSms.save()
                authSMS = AuthSMS.objects.create(profileUser=profile,
codeSended=sms_code, type_SMS=2, state_SMS=1)
                encode_information =
cryptografy.encodeAndSaveToken(user_id=profile.user.username,

password=profile.user.password, authSMS=authSMS,

state_SMS=1, time_expire_token=3, sms_code=sms_code)
                if encode_information[0]:
                    return JsonResponse({"message": "ok", "token":
encode_information[1]}, status=status.HTTP_200_OK)
                else:
                    return JsonResponse({"message": "service sms not
accesse,please try later time"},

status=status.HTTP_400_BAD_REQUEST)
            else:
                return JsonResponse({"message": "Duplicate code (or other
messages)"},
                                    status=status.HTTP_400_BAD_REQUEST)
        return Response({'success': "Failed"},
status=status.HTTP_400_BAD_REQUEST)
```

شکل ۳–۷)کد ارسال توکن ورود به برنامه

در مرحله بعد با فراخوانی api، login-step2 حاوی اطلاعات توکن و کد تائید پیامک شده، در صورت درست بودن و تمام نشدن مهلت توکن، یک توکن برای احرازهویت در apiها دریافت می‌کند.

توکن‌های ساخته شد برای پیامک به‌صورت زیر با رمزنگاری SHA-256 ساخته می‌شود و ذخیره می‌شود:

```
def encodeAndSaveToken(user_id,password,sms_code,authSMS,state_SMS,time_expire_token):
    try:
        dt = datetime.now() + timedelta(minutes=time_expire_token)
        rand_int = random.randint(10000000000000, 99999999999999)
        encoded_token = jwt.encode(
            {'user_id': user_id, 'rand_int': rand_int},
            str(sms_code),
            algorithm='HS256')
        authSMS.token = encoded_token
        authSMS.state_SMS = state_SMS
        authSMS.save()
        return True,encoded_token
    except:
        return False
```

شکل ۳–۸) کد ذخیره‌سازی توکن در سرور

و برای بررسی تمام نشدن مهلت زمانی آن از تابع زیر استفاده می‌شود.:

```
def decodeAndSaveStateSMS(token,authSMS):
    try:
        decode_token = jwt.decode(token, str(authSMS.codeSended), algorithms=["HS256"])
        return True
    except:
        return False
```

شکل ۳-۹) کد زمان بندی توکن

بقیه apiهارو به‌صورت مختصر توضیح داده می‌شود و در صورتت استفاده از فنّاوری خاصی شرح داده می‌شود.

logout،Api برای از بین بردن توکن احرازهویت کاربر می‌باشد.

Apiهای request-send-reset-pass/ ، recive-codesms-and-send-token ، change-password-with-token برای تغییر پسورد می‌باشد که دارای سه مرحله بوده که در مرحله اول کاربر request-send-reset-pass،api با اطلاعات شماره موبایل فراخوانی کرده برای او یک پیامک ارسال‌شده و در جواب یک توکن می‌گیرد در مرحله بعد recive-codesms-and-send-token را با اطلاعات توکن که در مرحله قبل دریافت کرده و کدتانید پیامک ارسال کرده یک توکن در جواب دریافت نموده حال با فراخوانی change-password-with-token به همراه توکن مرحله قبل و گذرواژه جدید و تکرار گذرواژه جدید رمز خود را تغییر می‌دهد.

check-token،Api برای چک نمودن این می‌باشد که توکن احرازهویت اعتبار دارد یا خیر.

api-token-auth،Api برای دریافت توکن احرازهویت به‌صورت سریع بدون طی دو مرحله وبرای تست می‌باشد.

باید توجه داشت که برای امنیت بیشتر کاربر فقط می‌تواند هم‌زمان فقط یک توکن دارد و به‌محض ورود دوباره توکن قبلی از بین می‌رود.

همچنین برای مدیریت کاربران و مسائل احرازهویت در REST-FULLاز rest_framework.authentication.TokenAuthentication:

```
#swagger
REST_FRAMEWORK = { 'DEFAULT_SCHEMA_CLASS': 'rest_framework.schemas.coreapi.AutoSchema',
                   'DEFAULT_AUTHENTICATION_CLASSES': [
                       # 'rest_framework.authentication.SessionAuthentication',
                       'rest_framework.authentication.BasicAuthentication',
                       'rest_framework.authentication.TokenAuthentication',

                   ],
                   }
```

شکل ۳-۱۰)شمای rest frame work

۳-۳-۲) Door Security : Urlهای مربوط به این App شامل زیر است:

```
urlpatterns = [
    path('open-door/', api.requestOpenDoor.as_view(), name='بازکردن درب'),
    path('add-member/', api.addMember.as_view(), name='اضافه کردن اعضا'),
    path('update-member/', api.updateMember.as_view(), name='تغییر اطلاعات فرد'),
    path('get-all-member-for-front/', api.getAllMember.as_view(), name='گرفتن تمام اعضا'),
    path('get-history/', api.getHistory.as_view(), name='تاریخچه'),
    path('add-history/', api.addHistory.as_view(), name='اضافه کردن ی تارخچه'),
    path('update-status-opendoor/', api.changeStatusOpenDoor.as_view(), name='عوض کردن وضعیت باز کردن درب'),
    path('get-member-for-rass/', api.getMembersForRassperyPi.as_view(), name='گرفتن ممبر برای rass'),
    path('delete-member/', api.deleteMember.as_view(), name='پاک کرد عوض'),
]
```

شکل ۳-۱۱) url های استفاده‌شده در door security

OpenDoor : برای باز کردن درب هست که کاربر با ارسال هش رزپری‌پای موردنظر خود درخواست باز کردن درب می‌کند درصورتی‌که رزپری‌پای به سرور وصل بود از طریق websocket پیامی برای رزپری‌پای ارسال می‌شود تا اقدام لازم را انجام دهد و درصورتی‌که آنلاین نبودن پیامی در همین منظور برای کاربر ارسال می‌شود.

Add Member : برای اضافه کردن عضو جدید برای باز کردن درب می‌باشد،برای این منظور اطلاعات هش رزپری‌پای ، نام شخص ، عنوان شخص ، base64 عکس فرد ارسال می‌شود.

Update Member : تغییر اطلاعات عضو می‌باشد که اطلاعات دریافتی مانند add-member می‌باشد به‌علاوه id عضو است.

Delete Member : برای پاک کردن یک عضو استفاده می‌شود که اطلاعات دریافتی شامل هش رزپری‌پای و id عضو می‌باشد.

get-all-member-for-front : برای دریافت اعضا سمت کاربر می‌باشد یک متد از نوع get بوده که با ارسال هش سریال رزپری‌پای ، یک لیست از اطلاعات عضوها که شامل idع عضو، عنوان،نام، عکس است را برمی‌گرداند.

get-all-member-for-rass : مانند get-all-member-for-front است با این تفاوت که جای هش سریال رزپری‌پای،سریال رزپری‌پای و توکن دریافت می‌کند.

در دو api، get-all-member-for-front و get-all-member-for-rass یک فیلد به نام id داریم که اگر مقدار آن 1- باشد همه اعضا را برمی‌گرداند در غیر این صورت اگر id برای عضو رزپری‌پای بود اطلاعات فقط آن فرد را برمی‌گرداند.

add-history : برای اضافه کردن یک تاریخچه جدید توسط دستگاه به کار می‌رود که با ارسال اطلاعات سریال رزپری‌پای و توکن ، اگر شخص را شناسایی کند اطلاعات فرد را به همراه وضعیت ، id فرد و عکس گرفته‌شده را ارسال می‌کند در غیر این صورت فقط وضعیت و عکس گرفته‌شده را ارسال می‌کند.

get-history : برای دریافت فهرستی از تاریخچه عبور و مرور می‌باشد.متد آن از نوع get بوده.با دریافت هش رزپری‌پای برمی‌گرداند.

update-status-opendoor : برای باز کردن درب بوده که کاربر با واردکردن هش رزپری‌پای دستگاه موردنظرش وضعیت امکان باز کردن درب را تغییر می‌دهد.

باید توجه داشت که همه apiهایی که برای کاربر می‌باشد باید فرد ابتدا login کرده سپس api موردنظر خودزا فراخوانی کند.

۳-۳-۳) general App :

این app شامل یک api زیر است:

```
app_name='generalApp'
urlpatterns=[
    path('get-information/',api.getInformations.as_view(),name='اطلاعات سرویس ها')
]
```

شکل ۳-۱۲)شمای general app

get-information : برای دریافت اطلاعات سیستم‌های کاربر است که بخشی از آن برای فراخوانی api موردنظر نیاز است و از نوع get بوده.اطلاعاتی که برمی‌گرداند شامل json از عنوان ،آدرس ، هش رزپری‌پای می‌باشد.

۳-٤) Middleware :

Middlewareها به‌کاررفته در پروژه: در app، doorSecurity T، middleware زیر وجود دارد:

۱- این middleware برای بررسی این هست که درخواست برای دستگاه خود می‌باشد یا خیر.

```
class checkHashRassSerial(MiddlewareMixin):
    WHITELISTED_URLS = [
        '/door-security/open-door/',
        '/door-security/get-all-member-for-front/',
        '/door-security/get-history/'
        '/door-security/update-member/',
        '/door-security/add-member/',
        '/generall/get-information/'
    ]
    def process_request(self, request):
        if request.path in self.WHITELISTED_URLS:
            try:
                body_unicode = request.body.decode('utf-8')
                body = json.loads(body_unicode)
                hash_serial_rasperyPi = body['hash_serial_rasperyPi']
                licenseToUse = InformationService.objects.get(
                    rassperypiInfo__hash_serial_rasperyPi=str(hash_serial_rasperyPi),
                    rassperypiInfo__profile__user=request.user).lincense
            except:
                return JsonResponse({"message": "no hash rassperyPi for you"},
                                    status=status.HTTP_404_NOT_FOUND)
            return None
        else:
            return None
```

شکل ۳-۱۳)کد middleware استفاده شده-شماره یک

در لیست WHITELISTED_URLS آدرس urlهایی که می‌خواهیم این middleware روی آن اجرا شود را قرار می‌دهیم.

در تابع process_request بررسی می‌کنیم که آدرسی که فراخوانی شده در لیست است یا خیر، اگر بود هش سریال رزپری‌پای را برداشته و با آن چک می‌کند که دستگاه برای فرد بوده است یا خیر.درصورتی‌که به فرد متعلق نباشد از ادامه کار جلوگیری کرده پیام مربوطه ارسال می‌شود با status 404.

۲- این middleware برای بررسی وجود لاینسس برای دستگاه می‌باشد.

```python
class exitLincenceMiddleware(MiddlewareMixin):
    WHITELISTED_URLS = [
        '/door-security/open-door/',
        '/door-security/get-all-member/',
        '/door-security/get-history/'
        '/door-security/update-member/',
        '/door-security/add-member/',
    ]
    def process_request(self, request):
        if request.path in self.WHITELISTED_URLS:
            try:
                body_unicode = request.body.decode('utf-8')
                body = json.loads(body_unicode)
                hash_serial_rasperyPi = body['hash_serial_rasperyPi']
            except:
                return JsonResponse({"message": "Duplicate code (or other messages)"},
                                    status=status.HTTP_400_BAD_REQUEST)

            try:
                licenseToUse=InformationService.objects.get(rassperypiInfo__hash_serial_rassperyPi=str(hash_serial_rasperyPi)).lincense
            except:
                return JsonResponse({"message": "no lincense for you"},
                                    status=status.HTTP_404_NOT_FOUND)
            return None
        else:
            return None
```

شکل ۳-۱٤)کد middleware استفاده شده-شماره دو

۳- این middleware برای بررسی این هست که id عضو که در api آمده است برای دستگاه موردنظر می‌باشد یا خیر.

```python
class checkMemberIsForRasspery(MiddlewareMixin):
    WHITELISTED_URLS = [
        '/door-security/update-member/',
        '/door-security/delete-member/',
    ]
    def process_request(self, request):
        if request.path in self.WHITELISTED_URLS:
            try:
                body_unicode = request.body.decode('utf-8')
                body = json.loads(body_unicode)
                hash_serial_rasperyPi = body['hash_serial_rasperyPi']
                id_member = body['id_member']
            except:
                return JsonResponse({"message": "Duplicate code (or other messages)"},
                                    status=status.HTTP_400_BAD_REQUEST)

            try:
                member = Members.objects.get(id=id_member, rassperySystem__hash_serial_rassperyPi=hash_serial_rasperyPi)
            except:
                return JsonResponse({"message": "no member exit for you"},
                                    status=status.HTTP_404_NOT_FOUND)
            return None
        else:
            return None
```

شکل ۳-۱۵)کد middleware استفاده شده-شماره سه

٤-این middleware برای بررسی منقضی نشدن لاینسس می‌باشد.

```python
class checkLincenseMiddleware(MiddlewareMixin):
    WHITELISTED_URLS = [
        '/door-security/open-door/',
        '/door-security/update-member/',
        '/door-security/add-member/',
    ]
    def process_request(self, request):
        if request.path in self.WHITELISTED_URLS:
            try:
                body_unicode = request.body.decode('utf-8')
                body = json.loads(body_unicode)
                hash_serial_rasperyPi = body['hash_serial_rasperyPi']
                licenseToUse = InformationService.objects.get(
                    rassperypiInfo__hash_serial_rassperyPi=str(hash_serial_rasperyPi)).lincense
                now = date.today()
            except:
                return JsonResponse({"message": "Duplicate code (or other messages)"},
                                    status=status.HTTP_400_BAD_REQUEST)

            if licenseToUse.end_lincense >= now:
                return None
            else:
                return JsonResponse({"message": "end lincence"},
                                    status=status.HTTP_404_NOT_FOUND)

        else:
            return None
```

شکل ۳-۱۶)کد middleware استفاده شده-شماره چهار

حال برای اینکه middlewareها به اجرا دربیاین باید آن‌ها را در settings سرور به‌صورت زیر اضافه کرد(چهار مورد آخر مربوط به middlewareهایی است که نوشته‌شده‌اند):

```python
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'doorSecurity.middleware.custom_middleware.exitLincenceMiddleware',
    'doorSecurity.middleware.custom_middleware.checkLincenseMiddleware',
    'doorSecurity.middleware.custom_middleware.checkMemberIsForRasspery',
    'doorSecurity.middleware.custom_middleware.checkHashRassSerial'
]
```

شکل ۳-۱۷)دستور اجرای middle ware ها

۳-۵) Websockets :

برای اطلاع‌رسانی به دستگاه در پروژه از websocket استفاده‌شده است زیرا با این کار نیاز نیست که دستگاه به‌صورت مکرر api را فراخوانی کند تا از تغییرات و پیغام‌های مربوط به خودآگاه شود بلکه با برقراری یک socket با سرور درصورتی‌که برای او پیغامی که به‌صورت دستوری می‌باشد ارسال می‌شود و دستگاه به‌صورت interup به آن را دریافت کرده و عمل لازم را انجام می‌دهد.

ابتدا برای آنکه سرور بتواند به websocket پاسخ بدهد لازم است تغییراتی در settings پروژه صورت گیرد:

۱- یک routing برای آدرس‌دهی برای websocket نیاز است که آن را در بخش اصلی پروژه به‌صورت فایل ایجاد کرده و در آن routing موردنظر را به‌صورت زیر قرار می‌دهیم:

```
from channels.routing import ProtocolTypeRouter,URLRouter
from channels.auth import AuthMiddlewareStack

from websocketManage import routing as doorSeurity_routing

application =ProtocolTypeRouter({
    'websocket': AuthMiddlewareStack(
        URLRouter(
            doorSeurity_routing.websocket_urlpatterns
        )
    )
})
```

شکل ۳-۱۸) routing برای آدرس‌دهی وب سوکت ها

۲- حال در app، webSocketManage یک فایل routingبرای url موردنیاز websocket ایجاد کرده و به consumer موردنظر اتصال می‌دهیم:

```
websocket_urlpatterns = [
    path('ws/open_door_websocket/',consumers.openDoorCunsumer.as_asgi(),name ='ساکت باز کردن درب')
]
```

شکل ۳-۱۹) اتصال وب سوکت به کاربر

۳- حال باید در settings موارد زیر اضافه شوند:

اضافه کردن ASWIبرای اشاره به routing موردنظر:

```
#channels
ASGI_APPLICATION = 'smart_video_door_phone.routing.application'
```

شکل ۳-۲۰)کد ارسال AWSI

مشخص کردن پایگاه داده موردنیاز websocket که در پروژه از همان پایگاه داده اصلی استفاده‌شده است:

```
CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'channels_postgres.core.PostgresChannelLayer',
        'CONFIG': {
            'ENGINE': 'django.db.backends.postgresql_psycopg2',
            'NAME': 'd805ekjcr8l9s6',
            'HOST':'ec2-54-211-160-34.compute-1.amazonaws.com',
            'PORT':5432,
            'USER': 'pjmrwufklrusgp',
            'PASSWORD': '4b378f4cd70405b3460ab0f9c3c7671460cdc480716b02c117982ed788006e67'

        },
    },
}
```

شکل ۳-۲۱) مشخص کردن پایگاه داده در وب سوکت

حال باید در فایل consumers در consumer،websocketManage موردنیاز را ایجاد کرد که یک کلاس با چهار تابع connect،disconnect، receive و یک تابع برای ارسال پیام به websocket موردنظر هست که نام آن را sendMassege گذاشته.

درصورتی‌که یک دستگاه به websocket وصل شد ابتدا تابع connect اجرا می‌شود که چک می‌گردد که سریال رزپری‌پای و توکن صحت دارند باهم در صورت صحت وضعیت آن دستگاه در دیتابیس آنلاین شده و گروهی بانام شامل سریال رزپری‌پای برای آن ایجاد می‌گردد:

```
def connect(self):
    self.check_level=False
    self.accept()
    try:
        self.token=str(dict(self.scope['headers'])[b'token'])
        self.token=self.token[2:len(self.token)-1]
        self.serial_rasperyPi = str(dict(self.scope['headers'])[b'serial-
rasperypi'])
        self.serial_rasperyPi =
self.serial_rasperyPi[2:len(self.serial_rasperyPi) - 1]
    except:
        self.send("not serial-rasperypi &| token")
        self.disconnect(close_code=1)
    else:
        try:

rassperypiInfo=RassperySystem.objects.get(serial_rasperyPi=self.serial_ra
sperypi , token_connect_rassperypi=self.token)
            rassperypiInfo.online_status=1
            rassperypiInfo.save()
            self.rassperypiInfo=rassperypiInfo
        except:
            self.send("not rassperypi with informathons")
            self.disconnect(close_code=1)
        else:
            self.rassperypiInfo.online_status=1
            self.rassperypiInfo.save()
            self.group_name=f"doorSecurity_{self.serial_rasperyPi}"
            async_to_sync(self.channel_layer.group_add)(
                self.group_name,
                self.channel_name
            )
            self.check_level = True

            try:

informationService=InformationService.objects.get(rassperypiInfo=self.ras
sperypiInfo)
                self.send(json.dumps({'massege': 'change status
requestOpenDoor', 'code': (1011 + informationService.status_opendoor)}))
            except:
                self.send("not rassperypi with informathons")
                self.disconnect(close_code=1)
```

شکل ۳-۲۲) کد اتصال به دستگاه

Disconnect برای زمانی است که ارتباط websocket قطع می‌گردد در این هنگام وضعیت دستگاه در پایگاه داده به آفلاین می‌رود و اطلاعات مربوط به اتصال آن از پایگاه داده قطع می‌گردد:

```
def disconnect(self, close_code):
    if self.check_level:
        self.rassperypiInfo.online_status=2
        self.rassperypiInfo.save()
        self.channel_layer.group_discard(
            self.group_name,
            self.channel_layer
        )
    else:
        self.close()
```

شکل ۳-۲۳) کد قطع دسترسی به دستگاه

تابع receive برای زمانی می‌باشد که پیامی از دستگاه آمد به آن پاسخ بدهد.

و تابع sendMassege برای ارسال پیام بوده:

```
def sendMassege(self, event):
    message=event['message']
    self.send(text_data=message)
```

شکل ۳-۲٤)تابع دریافت و ارسال

۳-۶) deploye کردن پروژه :

برای deploye کردن پروژه از heroku که به‌صورت رایگان با امکانات محدود سرور ارائه می‌دهد استفاده شد.

چالشی موجود عدم امکان استفاده از دو پورت از سرور بود زیرا در پروژه برای apiها و websocket برای هرکدام نیاز به یک پورت بود ولی heroku در نسخه رایگان فقط یک پورت ارائه می‌دهد.ازاین‌رو بعد از تحقیق و جستجو به کتابخانه‌ای از Django یافت شد که این امکان را ارائه می‌کرد برای هر دو آن‌ها از یک پورت استفاده شود و نام آن کتابخانه daphne بود.

برای deploye کرد در heroku نیاز است دو فایل procfile و requirements.txt ایجاد کرد.

در فایل procfile موارد زیر را نوشته:

```
release: python manage.py migrate
web: gunicorn smart_video_door_phone.asgi:application -k
uvicorn.workers.UvicornWorker
daphne: daphne smart_video_door_phone.asgi:application --port $PORT --
bind 0.0.0.0 -v2
```

شکل ۳-۲۵) اطلاعات درج‌شده در فایل‌های هیروکو

و در requirements.txt با دستور pip freeze > requirements.txt تمام کتابخانه‌های موردنیاز سرور را به‌صورت خودکار در آن نوشته تا در heroku در هنگام deploy آن‌ها را نصب نماید.

در settings نیاز است که تغییرات زیر صورت گیرد:

۱- ALLOWED_HOSTS = ['*']

۲-

```
STATIC_URL = '/static/'
import os
# Default primary key field type
# # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
VENV_PATH = os.path.dirname(BASE_DIR)
STATIC_ROOT = os.path.join(VENV_PATH, 'smart_video_door_phone/static_root')

MEDIA_URL='/media/'
MEDIA_ROOT=os.path.join(BASE_DIR,'media')
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

۳- حال در heroku یک new app کرده سپس آن را به github خود بخشی که پروژه در آن ریخته شده وصل نموده و آن را deploye کرده.

فصل چهارم

اپلیکیشن تلفن همراه

٤-١: نیازمندی‌ها و مفاهیم پایه

پس از طراحی و پیاده‌سازی بخش‌های قبل، نیاز به یک رابط کاربری مناسب جهت استفاده کاربران بود. به دلیل اینکه هدف طراحی این دستگاه استفاده در آپارتمان یا مجتمع‌ها است، باید رابط کاربری برای بیشتر سنین مناسب باشد و با توجه به اینکه اکثر مردم با اپلیکیشن هایی نظیر تلگرام و واتس اپ آشنایی دارند، بهترین انتخاب، اپلیکیشن تلفن همراه بود.

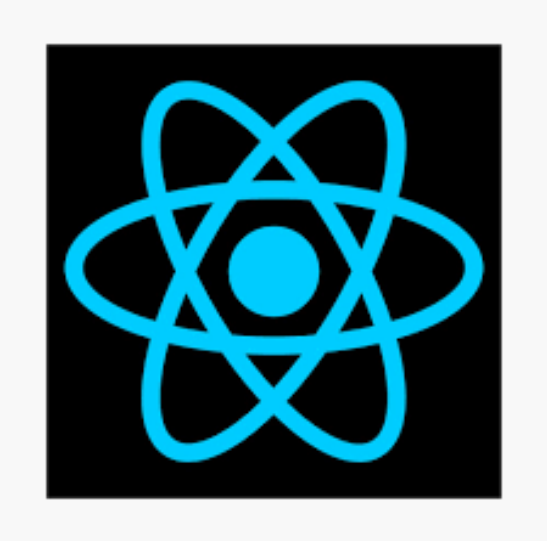
این برنامه توسط یک ادمین اداره می‌شود و باید اعضایی که اجازه ورود دارند را تعریف بکند و هم‌زمان قابلیت حذف یا اضافه کردن عضو جدید و همچنین نظارت بر دوربین را داشته باشد تا در صورت نیاز، به‌صورت دستی در ورودی را باز بکند.
دیگر نیازمندی این برنامه، این است که باید عکس، عنوان، نام و نام خانوادگی عضوها را داشته باشد و عکس ارسال‌شده به سرور، مستقیماً به دستگاه داده می‌شود تا پس از تأیید اطلاعات عضو، اجازه ورود به آن داده و در را باز بکند.

برای اینکه برنامه هم در اندروید و هم در ios در دسترس باشد باید زبانی انتخاب می‌شد تا هر دو خروجی را به ما بدهد؛ ازاین‌جهت به زبانی native موردنیاز بود که درنهایت به دلیل تجربه کاری، react nativeبه‌عنوان زبان‌اصلی انتخاب شد.
Native app development، روندی برای پیاده‌سازی برنامه در بسترهای android، ios و وب هست و خروجی برنامه را در محیط‌های ذکرشده در دسترس قرار می‌دهد.

از فنّاوری‌های به‌کاررفته می‌توان به موارد زیر اشاره کرد:

React native: این زبان بهترین بخش‌های native را با بستر react js ترکیب می‌کند و می‌توان با استفاده از آن برنامه‌های تحت وب، اپلیکیشن android و ios پیاده‌سازی کرد.

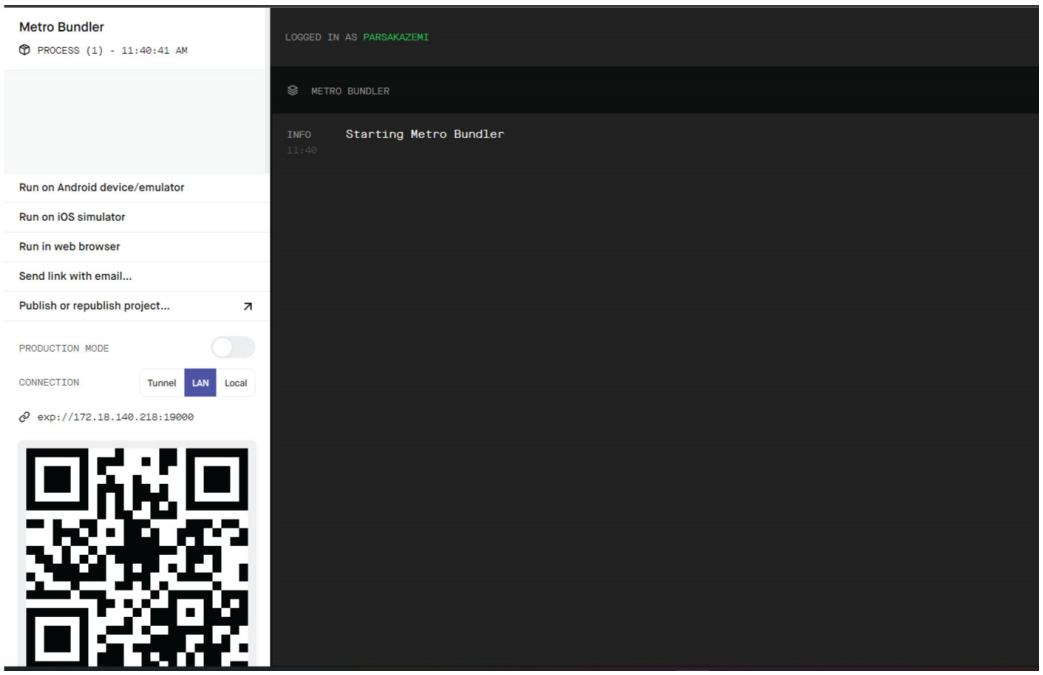
React: یک کتابخانه JavaScript front-end رایگان برای ساخت رابط‌های کاربری بر اساس UI component است که می‌توان از آن در ساخت برنامه‌های تک‌صفحه‌ای، اپلیکیشن، وب‌سایت و غیره استفاده کرد و به‌جای اجرا برنامه سمت سرور و اعلام نتیجه بر روی رابط کاربری، برنامه را در سمت کاربر اجرا می‌کند که درنتیجه سرعت و عملکرد سرور را افزایش می‌دهد و با کمک افزونه JavaScript Syntax Extension یا jsx می‌تواند کدی مشابه html تولید کند.



شکل ٤-١) لوگو زبان react

Expo: مجموعه ایی از ابزارهای ساخت برنامه react native هست که روند ساخت و نشر برنامه را آسان‌تر می‌کند و با امکاناتی نظیر شبیه‌ساز تحت وب و ایجاد دمو آنلاین، امکان آزمودن برنامه را به ما ارائه می‌دهد.

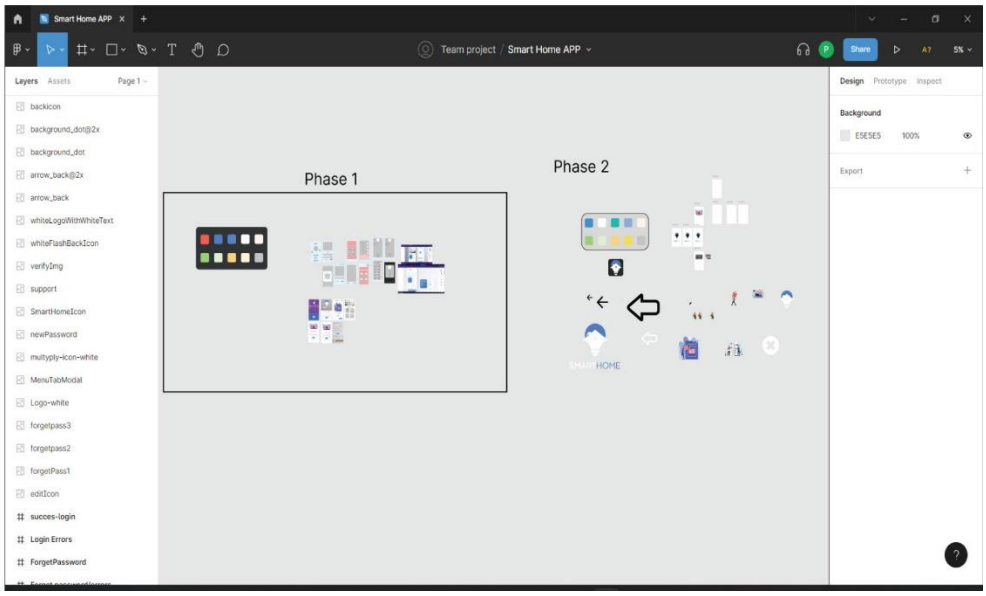
کار با آن مشابه کار با react native هست و با کنسول قابل انجام است. درنتیجه بهترین روند پیاده‌سازی با این زبان، استفاده از expo یا expo cli است.



شکل ٤-٢) شمای ابزار expo که در آن روش‌های مختلف اجرا دمو و ساخت برنامه را ارائه می‌دهد.

Figma: برنامه‌ای تحت وب برای پیاده‌سازی سند رابط کاربری است و امکان مشاهده آنلاین و لحظه‌به‌لحظه پروژه را به اعضا گروه می‌دهد. با طراحی قبل از پیاده‌سازی رابط کاربری هرگونه تغییر در رابط کاربری یا نیازمندی‌ها، وقت برنامه‌نویس را نگرفته و با امکاناتی مثل **prototype**، پیاده‌سازی را راحت‌تر می‌کند.

Prototype ابزاری برای ایجاد یک دمو برای رابط کاربری در برنامه figma است که تقریباً بیشتر نیازمندی‌ها و امکانات برنامه نهایی را بدون برنامه‌نویسی و عملکرد واقعی، پیاده‌سازی می‌کند و می‌توان قبل از شروع کد نویسی، جزییات هر فعالیت و حتی پویانمایی صفحات را دید و به‌نوعی یک road map در اختیار برنامه‌نویس قرار می‌دهد.

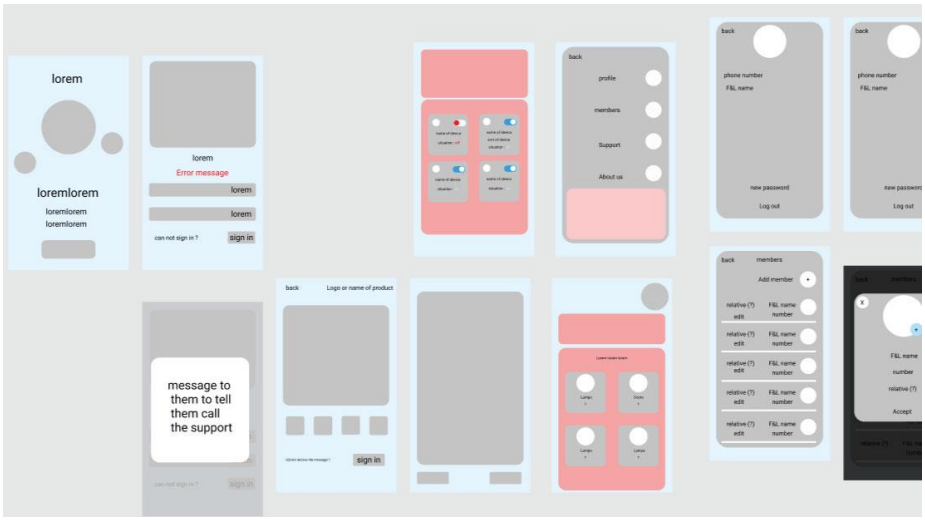


شکل ٤-٣) شمای برنامه figma جهت ساخت داکیومنت های گرافیکی ui/ux

٤-٢: طراحی داکیومنت رابط کاربری (UI/UX)

قبل از پیاده‌سازی برنامه، ابتدا باید به یک قالب گرافیکی موردقبول دست پیدا می‌کردیم تا در صورت تغییر ناگهانی یا اضافه شدن نیازمندی‌های جدیدتر، پیاده‌سازی برنامه دچار مشکل نشود. درنتیجه با figma، داکیومنتی طراحی شد تا برنامه‌نویس با کمک آن اقدام به پیاده‌سازی برنامه بکند.

ابتدا شمای خطی یا wire frame ای برنامه تهیه می‌شود و در آن تمامی امکانات و رنگ‌های استفاده‌شده در برنامه بدون در نظر گرفتن هرگونه جزییات از قبیل عکس، اکشن‌های کاربر و پویانمایی‌ها را به نمایش می‌گذارد.



شکل ٤-٤) نمونه ایی از wire frame که بدنه‌ی صفحات بدون ذکر جزییات را نمایش می‌دهد.

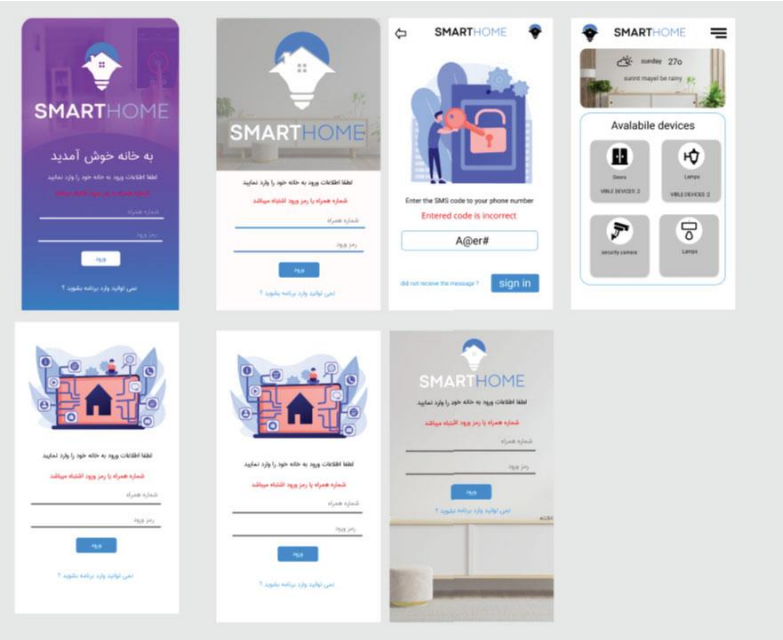
پس‌ازاینکه تمام نیازمندی‌ها در شمای خطی پیاده‌سازی شد جزییات هر صفحه به‌صورت جداگانه تعریف می‌شود. این جزییات در آینده توسط برنامه‌نویس پیاده‌سازی می‌شود. به این مجموعه جزییات prototyping گفته می‌شود که به‌عنوان نمونه عبارت‌اند از:

– انواع اکشن‌هایی که کاربر در هر صفحه به‌صورت جداگانه داشته باشد؛ مثل کلیک بر روی دکمه، آپلود عکس و ... .

–انواع مختلف انیمیشن‌هایی که بین صفحات یا قبل و بعد هر اکشن ممکن است نیاز باشد مثل انیمیشن انتظار برای پاسخ سرور در صفحه ورود به برنامه، انیمیشن انتقال به یک صفحه یا برگشت به آن و ...

–جایگذاری انواع رنگ‌های موردنیاز در مکان‌های در نظر گرفته‌شده، جایگذاری عکس‌های نمونه که باید شبیه به محصول نهایی باشند و ... .

پس از ساخت prototype، داکیومنت برای پیاده‌سازی به دست برنامه‌نویس می‌رسد.



شکل ٤–٥) نمونه ایی از طرح خطی که جزییاتش کامل شده است (این صفحات نمونه‌های کلی از برنامه هستند ه درنهایت یکی از آن‌ها به‌عنوان نسخه نهایی انتخاب می‌شود)

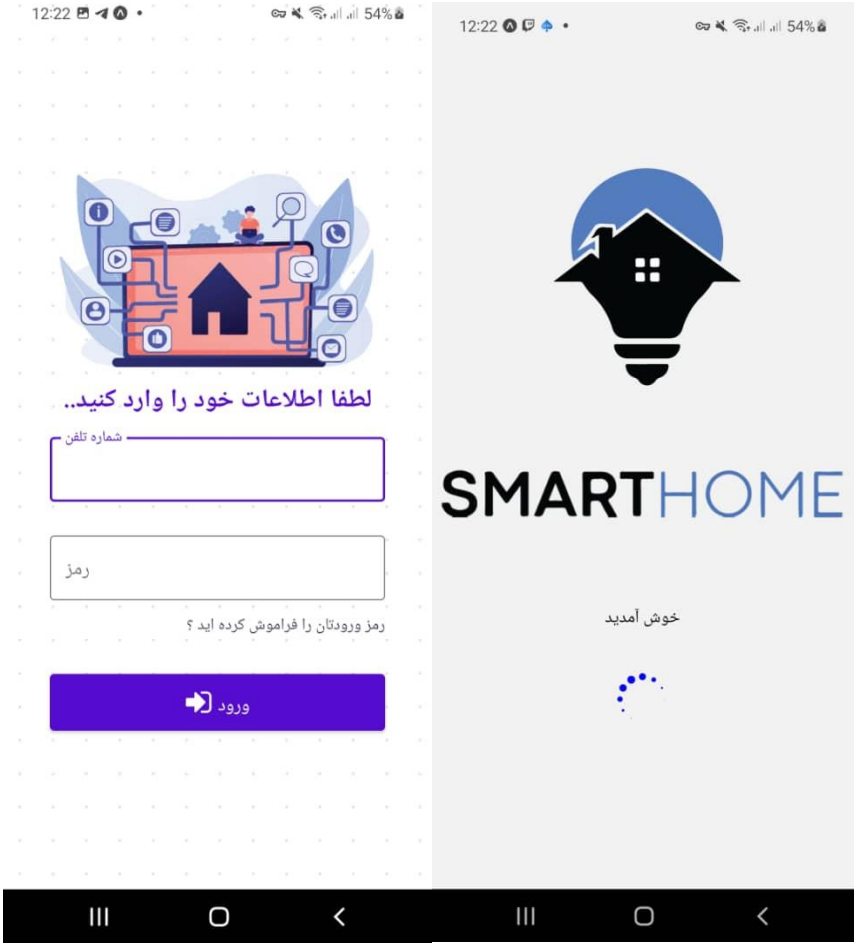
٤-٣: پیاده‌سازی برنامه

پس از دسترسی به داکیومنت رابط کاربر، روند پیاده‌سازی برنامه شروع می‌شود. بر اساس داکیومنت، صفحات به ترتیب پیاده‌سازی می‌شوند ولی در این مرحله به سرور متصل نمی‌شود. ابتدا صفحه ورود به سیستم که شامل فرم اطلاعات ورودی و فراموشی رمز است.

صفحات مربوطه طبق داکیومنت تنها پیاده‌سازی شدند و بعد از اعمال شدن استایل و مدل چینش اجرا آن‌ها، روند متصل کردن برنامه به سرور شروع می‌شود.

به دلیل اینکه از قبل در داکیومنت صفحات در دسترس است، نیاز به پیاده‌سازی گراف انتقال داده نیست و انتقال داده به‌جز چند مورد به‌راحتی صورت می‌گیرد و با کمک سازنده سرور، روند اتصال صورت می‌گیرد.

شکل ٤-٦) صفحه‌ی تاریخچه ورود اعضا



شکل ٤-٧ و ٤-٨) صفحات انتظار برنامه (راست) و ورود اعضا (چپ)

متصل شدن به سرور، مجموعه ایی از درخواست‌هایی است که کاربر برای برطرف کردن نیازمندی‌هایش به سرور ارسال می‌کند و منتظر پاسخ می‌ماند. این کار در react native با دستور fetch انجام می‌شود، ولی به دلیل اینکه دستور fetch پیاده‌سازی طولانی با جزییات زیاد است، از پکیج axio استفاده‌شده است که پیچیدگی دستور fetch را از بین برده است و درخواست از سرور و دریافت آن را ساده‌تر کرده است.

درنهایت پس‌ازاینکه تمام درخواست‌ها طبق نیازمندی‌های برنامه نتیجه مطلوبی داد، روند ساخت اپلیکیشن، تمام می‌شود.

٤-٤: تست و انتشار

پس از پایان روند پیاده‌سازی اپلیکیشن، نیاز به آزمودن آن در دستگاه‌های مختلف و منتشر کردن آن در بسترهایی است که کاربران بتوانند از آن استفاده کنند. یکی از امکانات expo این است که پروژه را بدون ساخت نهایی، می‌توان منتشر کرد و کاربران یا تسترها از طریق یک لینک آنلاین، می‌توانند دمو آنلاین برنامه را ببینند و مشکلات آن را گزارش بدهند.

پس از آزمودن برنامه و پاسخ به بیشتر نیازمندی‌ها، به مرحله انتشار نهایی رسیدیم ابزار expo پس‌ازاینکه لینک آنلاینی برای دمو برنامه در اختیار ما قرار می‌دهد، نمونه ایی از نسخه نهایی برنامه را هم آماده‌سازی می‌کند تا در صورت تأیید دمو، فایل نهایی آن را بسازد و در دستگاه ذخیره کند.

خروجی برنامه می‌تواند فایل‌های apk و ios باشد. با دریافت این دستگاه، اپلیکیشن و اطلاعات لازم برای ورود را دریافت می‌کند و با نصب آن در دستگاه به امکاناتش دسترسی خواهد داشت...

شکل ٤-٩) دستورات در کنسول برای ساخت یا اجرای دمو برنامه

فصل پنجم

پردازش تصویر دستگاه

۵-۱) نیازمندی های پردازش تصویر :

هر سخت‌افزاری، به یک نرم‌افزار یا برنامه جهت اجرا نیاز دارد و پس‌ازاینکه دستگاه ما آماده شد، منتظر برنامه ایی بود تا کار خود را شروع کند. در این بخش با فناوری پردازش تصویر و مواردی که در ادامه آن‌ها را موردبررسی قرار می‌دهیم، برنامه ایی پیاده و به سخت‌افزار داده شد.

روند کار این نرم‌افزار به این صورت است که:

۱ – تشخیص داده دریافت شده از دوربین و انجام پردازش تصویر، به‌منظور آماده شدن برای تشخیص هویت چهره.

۲ – بررسی چهره داخل عکس و بررسی واقعی یا عدم واقعی بودن تصویر و اجرای الگوریتم هوش مصنوعی جهت شناسایی هویت چهره.

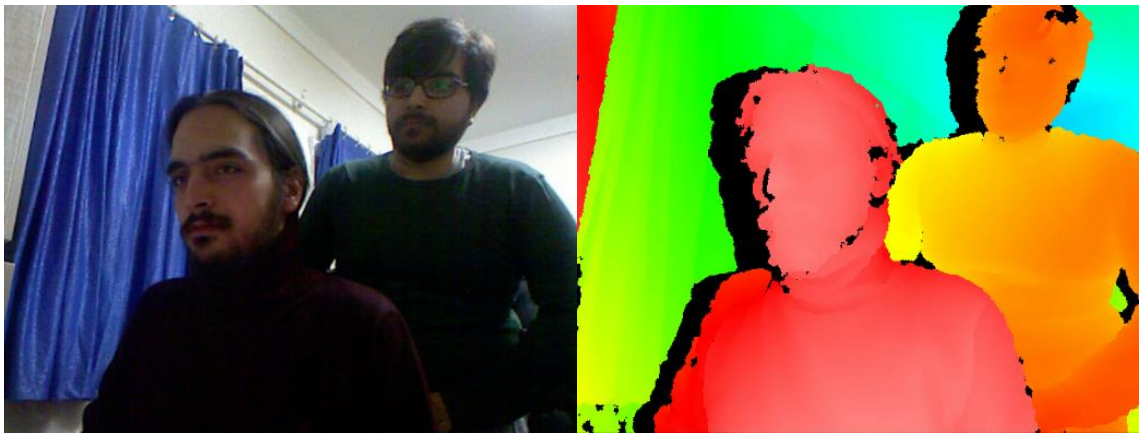
۳ – ارسال دستور باز شدن یا نشدن به رله دربازکن.

٤ – ارسال دستور روشن شدن دیود موردنظر.

۵ – اجرای یک کد به‌صورت socket programming جهت کنترل و مدیریت دربازکن از طریق ارتباط با یک اپلیکیشن تلفن همراه.

۵-۲) روند پردازش تصویر :

دریافت تصویر از دوربین: از طریق دوربین TOF یا همان دوربین kinnect، دو نوع تصویر؛ RGB و Depth map دریافت می‌شود.



شکل۵-۱ و ۵-۲) تصویر RGB دریافت شده از دوربین (سمت چپ)، تصویر depth map دریافت شده از دوربین (سمت راست)

زبان برنامه‌نویسی جهت پیاده‌سازی این هدف، پایتون انتخاب شد که یک‌زبان مفسری، باقابلیت بسیار بالا و کتابخانه‌های مختلف و کاربردی درزمینهٔ هوش مصنوعی، ساخت مدل شبکه‌های عصبی و همچنین کتابخانه‌هایی به‌منظور روندهای تحت وب برای ارتباط با اپلیکیشن تلفن همراه است.

برای تشخیص چهره واقعی از چهره غیر واقعی با استفاده از تصویر عمقی گرفته‌شده از دوربین، ابتدا نیاز است که از طریق کتابخانه opencv در پایتون، یک مدل شبکه‌های عصبی casscade بسازیم.

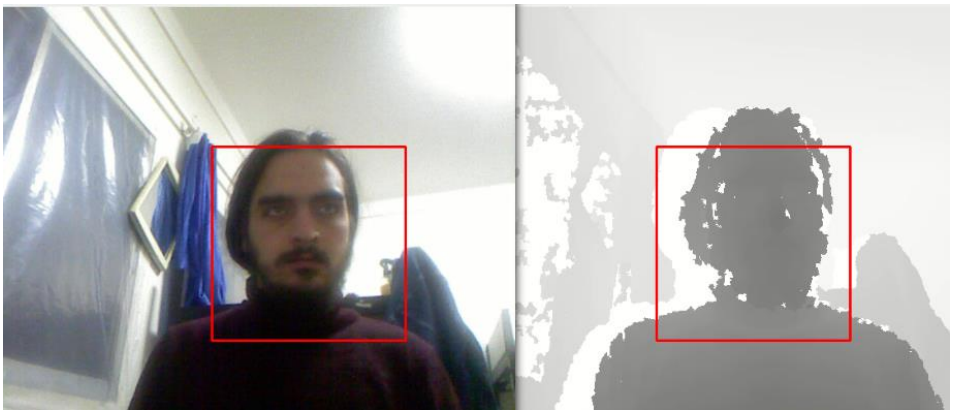
بعد از جمع‌آوری مجموعه داده مناسب از طریق دوربین و دسته‌بندی کردن آن‌ها به دودسته؛ عکس‌هایی با چهره‌های واقعی و دسته دیگر عکس‌هایی که در آن‌ها تصویر واقعی صورت وجود ندارد، آن‌ها را به برنامه پردازش تصویر داده تا رود یادگیری را انجام دهد.

```
- faces/
  Training Cascade Classifier/negative
  Training Cascade Classifier/positive
  Training Cascade Classifier/face_annotation
```

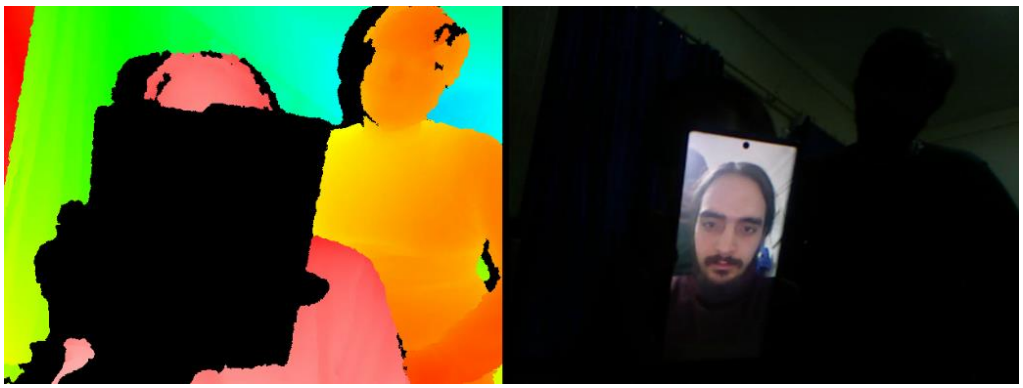
```
1   Training Cascade Classifier/negative
2   Training Cascade Classifier/positive
3   Training Cascade Classifier/face_annotation
```

شکل۵-۳) روند یادگیری مجموعه داده

پس از یادگیری، مدل آن را بر روی تصویر گرفته‌شده اعمال می‌کنیم که در صورت تشخیص چهره واقعی، یک مربع قرمزرنگ به دور آن بکشد و در صورت تشخیص چهره غیرواقعی، هیچ عملی انجام ندهد.

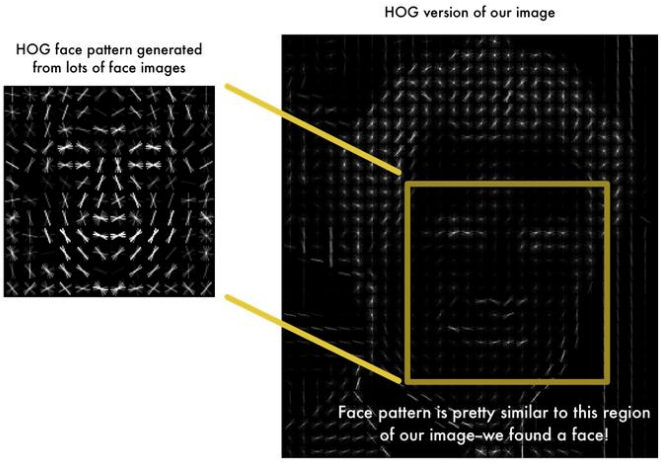


شکل۵-٤) خروجی تشخیص یک چهره واقعی توسط برنامه

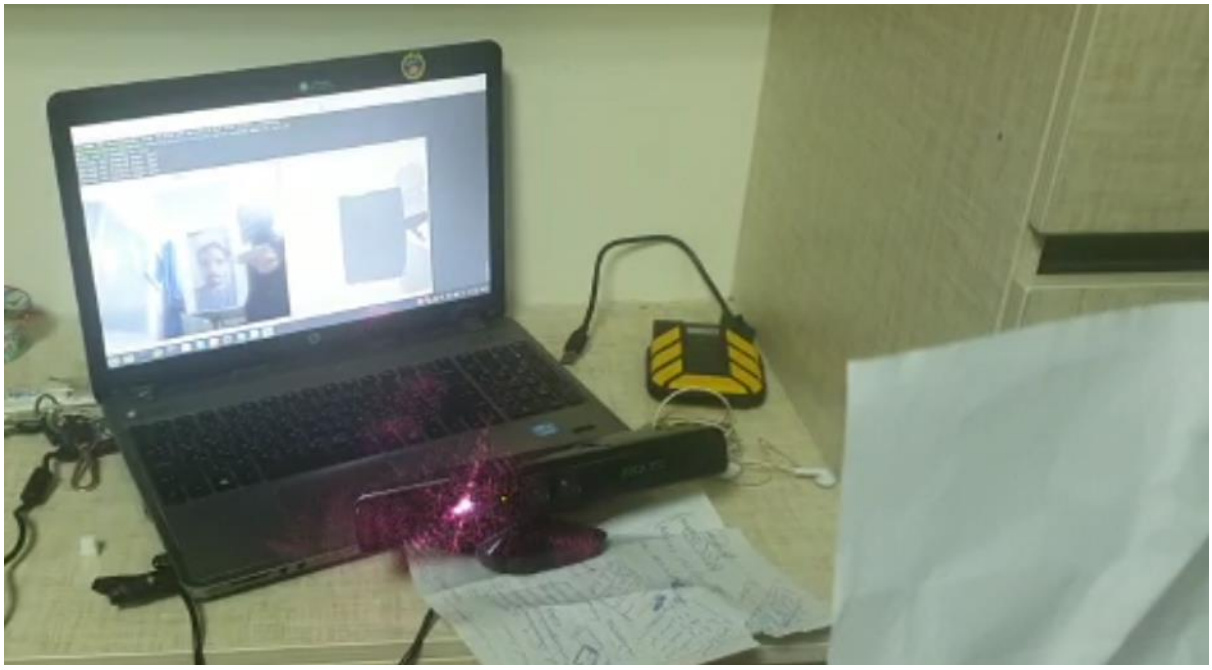


شکل ۵-۵) عکس چهره غیرواقعی (سمت راست)، خروجی عدم‌تشخیص چهره توسط دستگاه (سمت چپ)

پس از تشخیص چهره، به یک مدل برای شناسایی هویت تصویر نیاز است. تصویر را به یک مدل که منظور face authorization استفاده می‌شود، می‌دهیم. برای این کار، از یک مدل آماده کتابخانه opencv استفاده می‌کنیم تا نوع آن را به hnn تبدیل کند و بدین‌صورت، با استفاده از GPU کمتر می‌توانیم سرعت تشخیص را بالاتر ببریم.



شکل ۵-۶) نسخه HOG عکس گرفته‌شده به همراه الگو چهره



شکل ۵-۷) عدم‌تشخیص چهره واقعی توسط دوربین

سپس با استفاده از کتابخانه‌های GPIO در رزبری پای که برای کنترل پایه‌های خروجی و ورودی آن می‌باشد، پایه‌های رله و دیودها را کنترل می‌کنیم.

شکل ۵-۸) عدم‌تشخیص چهره واقعی که چراغ‌قرمز برگردانده است

شکل۵-۹) تشخیص چهره واقعی که چراغ سبز برگردانده است

در بخش دیگر پروژه نیاز داریم تا با socket programming، برنامه اجرا شود و با اپلیکیشن تعامل داشته باشیم و بدون نیاز به نگه‌داشتن خود برنامه در کنار برنامه به‌صورت یک thread، در کنار آن اجرا شود و بتواند نیازمندی‌های **اضافه شدن کاربر جدید، دستور باز شدن**،متوقف کردن **دستگاه از شناسایی و باز کردن در** را برطرف کند.

بخش دوم

سخت افزار

آنچه در بخش سخت افزار مطالعه میکنید :

درباز کن هوشمند شامل دو بخش نرم افزار و سخت افزار است که تا اینجا ، بخش نرم افزار آن مورد مطالعه قرار گرفت.این نرم افزار ها همگی بر اساس دستگاهی پیاده سازی شده اند که انها را اجرا کند و بین سرور ،دوربین ، اپلیکیشن ارتباط ایجاد کند.

دستگاه باید شکلی مشابه دیگر دربازکن ها داشته باشد ، ولی به دلیل اینکه یک رایانه مرکزی باید اطلاعات را بفرستد و دریافت کند ، شبیه سازی شده ساختمان در این پروژه پیاده سازی شده است ؛ به گونه ایی که در دنیای واقعی ، تنها دوربین بر روی دستگاه نصب میشود و کامپیوتر مرکزی با فاصله دورتری در جای امن است ، ولی در این قدم از پیاده سازی ، همه اجزا در کنار هم به صورت فشرده پیاده سازی شده است.

در این بخش به اجزا تشکیل دهنده سخت افزار مثل ، دوربین ، دربازکن ، مدارهای به کار رفته در دستگاه ، رله و ... میپردازیم.

فصل ششم

پیاده سازی دستگاه

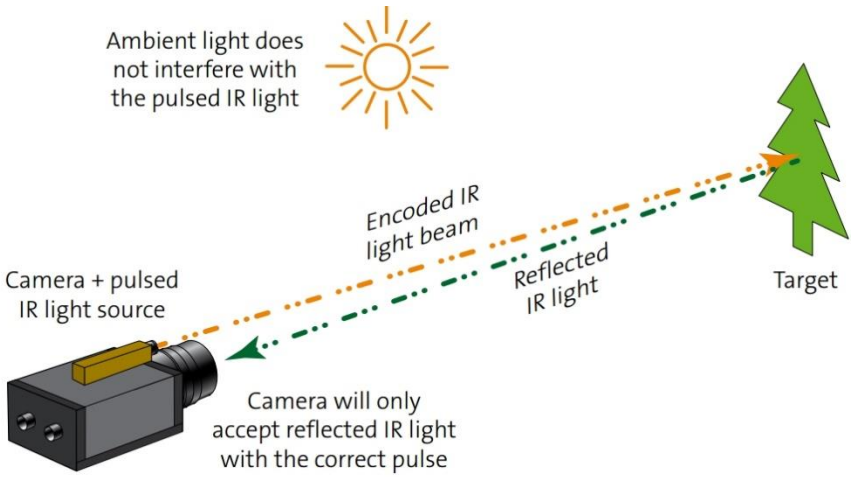
۵-۱) نیازمندی‌ها و مفاهیم پایه

برای ساخت دستگاه دربازکن هوشمند نیاز به سخت‌افزاری بود که بتواند برنامه تشخیص چهره و ارسال دستور باز شدن در به دربازکن را صادر کند لذا اقدامات لازم جهت فراهم کردن نیازمندی‌ها انجام شد از قطعات مهم مورداستفاده در بخش سخت‌افزار می‌توان به موارد زیر اشاره کرد:

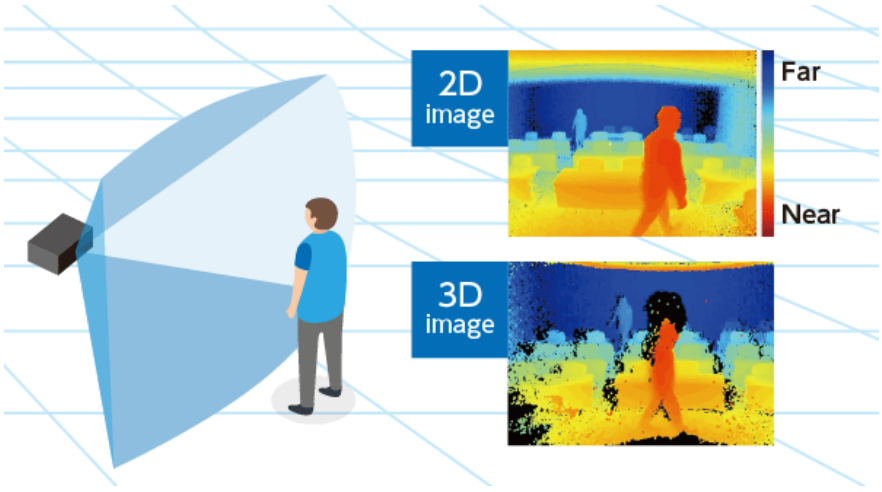
Rasberry Pi: رزبری پای نوعی میکروپروسسور و یا نوعی کامپیوتر کوچک است که برای اجرا کردن کد از آن استفاده می‌شود. البته یکی از ویژگی‌های خوب این قطعه ابعاد و اندازه آن است که به علت داشتن سایز کوچک بسیار برای این پروژه مناسب بود. رزبری پای مورداستفاده در این دستگاه، رزبری پای ٤ مدل B هست که با رم ۸ مگابایتی خود یکی از بهترین مینی کامپیوترهایی است که می‌توان از آن استفاده کرد.

شکل۵-۱) نمای رزبری پای ٤ مدل B

دوربین TOF: این دوربین که مخفف کلمه time of flight است یک نوع دوربین برای تصویربرداری عمقی از شیءها می‌باشد. نحوه کارکرد آن بدین شکل است که با ارسال تعدادی امواج رادیویی و محاسبه سرعت بازتاب آن‌ها به حس‌گر گیرنده، فاصله نقاط مختلف از هم را اندازه‌گیری می‌کند و در ادامه تصویری به‌صورت تصویر عمقی از شیء که در مقابل عدسی آن قرار دارد تشکیل می‌دهد؛ لذا این دوربین برای اسکن کردن اشیا سه‌بعدی بسیار مناسب است.



شکل ۵-۲)پرتاب امواج رادیویی توسط دوربین tof



شکل ۵-۳) دریافت خروجی دوربین tof

تشخیص این‌که آیا چهره قرارگرفته در مقابل دوربین یک عکس از شخص است یا خود واقعی شخص، به روش زیر انجام می شود:

اگر تصویر شخص در مقابل دوربین باشد تصویر قرارگرفته شده یک تصویر صورت است و ۲ بعدی می‌باشد لذا دوربین متوجه عدم واقعی بودن تصویر شده و عملیات باز کردن در را انجام نمی‌دهد اما درصورتی‌که چهره واقعی به‌صورت یک شیء سه‌بعدی باشد دستگاه شروع به شناسایی و تشخیص هویت شخص می‌کند که در صورت تأیید در را برای او باز می‌کند.

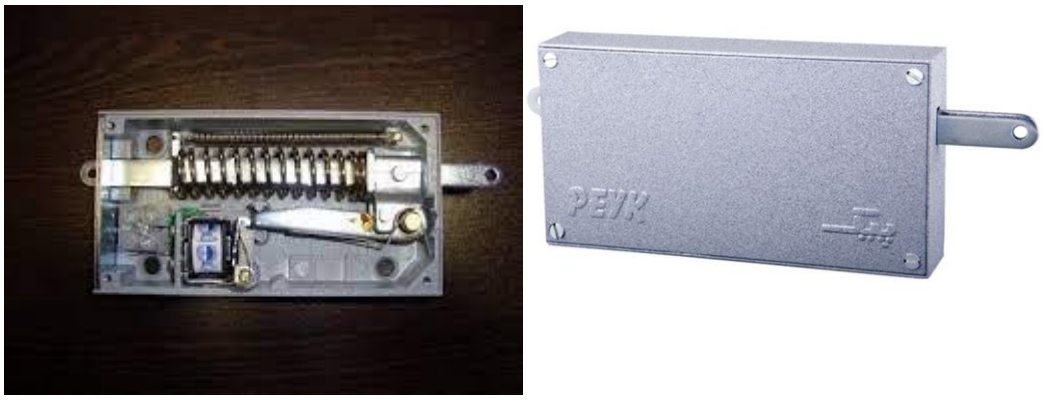
شکل۵-٤) دوربین tof

در این پروژه به علت مشکلات زیاد در تهیه دوربین tof، از دوربین جایگزینی استفاده شد که تا حدودی همین کارایی را دارد، دوربین استفاده‌شده،kinnect، ساخت شرکت مایکروسافت است و برای کنسول‌های بازی (xbox 360) ساخته‌شده. دلیل این جایگزینی هم کم‌هزینه‌تر بودن و همچنین داشتن فناوری موردنیاز برای پیاده‌سازی این هدف می‌باشد.

شکل ۵-۵) کینکت ۳۶۰

دوربین RGB: این دوربین که همان کارایی دوربین معمولی را دارد و برای دریافت چهره رنگی استفاده می‌شود تا بعد از عملیات شناسایی چهره واقعی از چهره غیرواقعی، برای تشخیص هویت مورداستفاده قرار گیرد و در صورت تأیید هویت شخص، در را برای ایشان باز می‌کند که در این پروژه از دوربین کینکت این قابلیت را هم در اختیار ما قرار داده است.

دربازکن: یک جعبه کوچک استیل است که برای باز کردن در از آن استفاده می‌شود. مکانیزم درون آن به این شکل است که در صورت وصل شدن جریان برق، یک اهرم زنجیر متصل به زبانه در را می‌کشد تا درباز شود.



شکل ۵-٦ و ۵-۷) نمای بیرونی دربازکن (سمت راست)، اجزا داخلی دربازکن (سمت چپ)

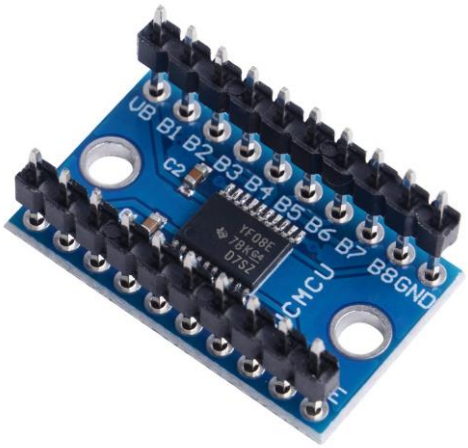
منبع تغذیه ۲۲۰ ولت به ۱۲ ولت: این منبع تغذیه وظیفه فراهم کردن برق دربازکن را دارد. دربازکن به یک برق ۱۵ ولت برای شروع کارش نیاز دارد که این منبع تغذیه با وصل شدن به برق شهری برق ۲۲۰ ولت ac را به برق ۱۲ ولت ac تبدیل می‌کند.

شکل ۵-۸) منبع تغذیه ۲۲۰ ولت به ۱۲۰ ولت

رله رزبری پای: رله یک قطعه الکترونیکی است که به‌عنوان کلید از آن استفاده می‌شود. یک سر آن به رزبری پای جهت ارسال دستور باز شدن به رله صادر می‌کند که رله با دریافت آن جریان براق را از دو سرخود عبور داده و جریان ۱۵ ولت تولیدشده توسط منبع تغذیه را به دربازکن وصل می‌کند و دربازکن، در را باز می‌کند.

شکل ۵-۹) عکس رله

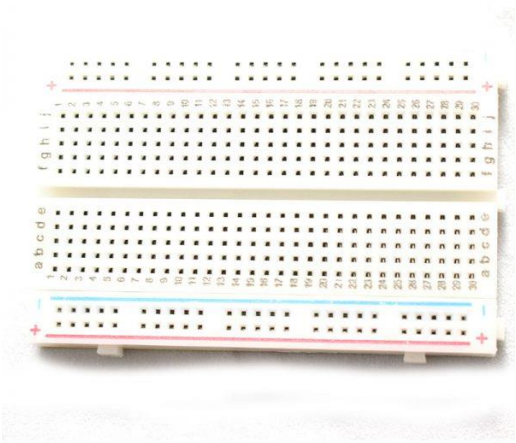
تبدیل‌کننده ۳ ولت به ۵ ولت: این‌یک مبدل برای تبدیل جریان ۳ ولت به ۵ ولت است، ازآنجاکه خروجی‌های رزبری پای، ۳ ولتی می‌باشد ولی ورودی رله ۵ ولت، از آن به‌منظور تبدیل جریان به ولتاژ مورداستفاده قرارگرفته است.



شکل ۵-۱۰) تبدیل‌کننده ۳ ولت به ۵ ولت

دیود: ۲ عدد دیود قرمز و آبی‌رنگ به‌منظور انتقال خروجی نتیجه اجرای عملیات تشخیص هویت توسط رزبری پای تعبیه‌شده که در صورت هرگونه خطا در شناسایی هویت دیود قرمزرنگ و در صورت موفقیت‌آمیز بودن دیود آبی‌رنگ روشن می‌شود.

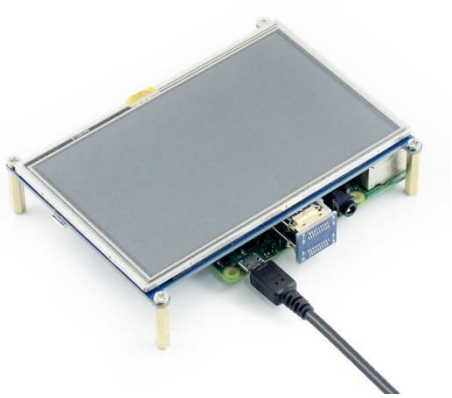
برد برد: از برد برد برای اتصالت موردنیاز از رزبری پای به led ها و تبدیل‌کننده ۳ ولت به ۵ ولت و همچنین رله استفاده می‌شود.



شکل ۵-۱۱) برد برد

نمایشگر: از یک نمایشگر برای نمایش تصویر قرارگرفته شده در مقابل دوربین و رابط کاربری بهتر استفاده است، صفحه نمایشگر مورداستفاده مانیتور رزبری پای ۵ اینچی می‌باشد.

دایرکتور برق: جعبه‌ای با جنس‌های مختلف ازجمله پلاستیکی و فلزی است که در این پروژه از دایرکتور برق پلاستیکی به علت کاهش وزن دستگاه استفاده‌شده است.



شکل ۵-۱۲) دایرکتور برق پلاستیکی

۵-۲) روند ساخت سخت‌افزار

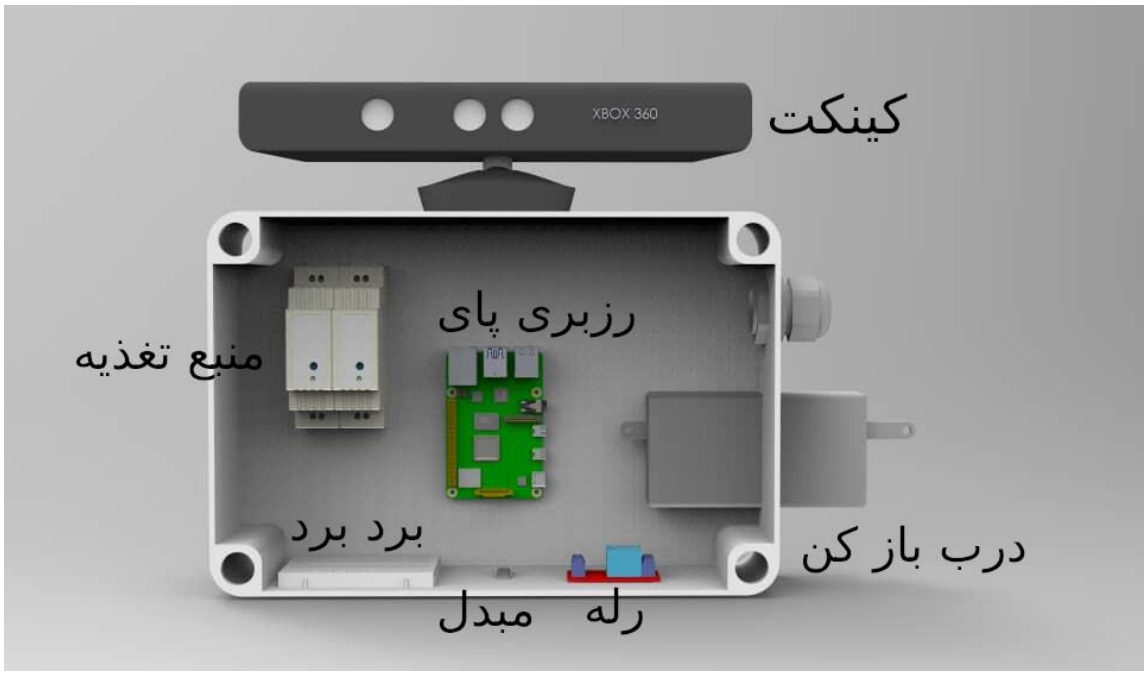
مرحله اول، مشخص کردن جای هر قطعه و طراحی مدار موردنیاز است تا بهترین مکان ممکن برای هر قطعه در نظر گرفته شود تا حتی‌الامکان پیچیدگی و در گسیختگی سیم‌های داخل دستگاه پیش نیاید و از کمترین جا، حداکثر استفاده را بتوان کرد.

مرحله دوم، برش‌های لازم جهت قرارگیری دیود و صفحه نمایشگر و سوراخ‌هایی برای گذر دادن سیم‌های خروجی اعم از سیم‌های جریان برق و محل قرارگیری پیچ‌ها انجام شد.

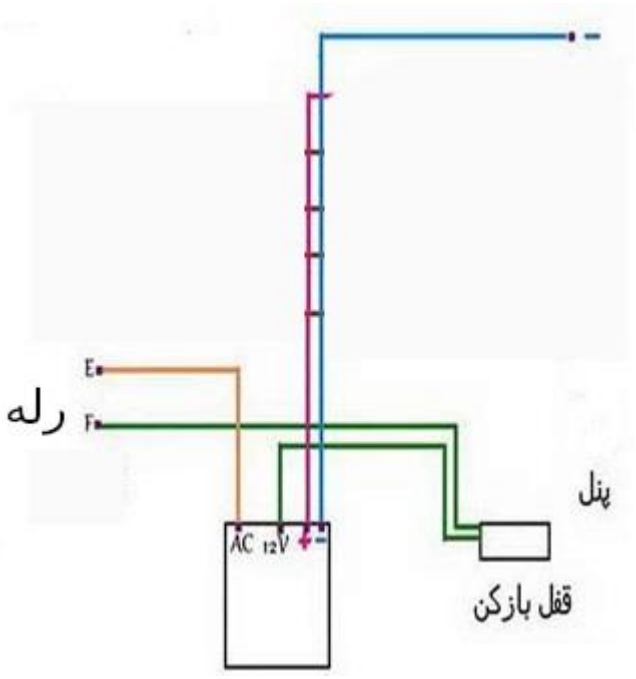
محله سوم، نصب قطعات در مکان طراحی‌شده خود، انجام شد و سیم‌کشی‌های لازم صورت گرفت.

مرحله چهارم، کد برنامه نوشته‌شده بر روی رزبری پای پیاده‌سازی و اجرا شد.

شکل۵-۱۳) نمای جلو دربازکن هوشمند



شکل ۵-۱٤) نمای داخلی دستگاه دربازکن هوشمند



شکل ۵-۱۵) مدار بکار رفته در برد برد دستگاه

پیشنهادها

برای جزییات بیشتر در مورد پردازش تصویر به آدرس‌های زیر مراجعه شود:

Digital Image Processing

برای آگاهی از روند زبان‌های native به آدرس‌های زیر مراجعه شود:

Choosing the Best Programming Language for Your Native App

برای آشنایی با UI/UX:

What is UI design? What is UX design? UI vs UX

فهرست منابع

1. https://reactnative.dev/docs/getting-started

2. https://docs.expo.dev/

3. https://designlab.com/figma-101-course/introduction-to-figma/

4. https://docs.opencv.org/4.x/

5. https://www.javatpoint.com/opencv

6. https://tutorials-raspberrypi.com/raspberry-pi-control-relay-switch-via-gpio/

7. https://openkinect.org/wiki/Getting_Started

8. https://www.kdab.com/setting-up-kinect-for-programming-in-linux-part-1/

9. https://www.terabee.com/shop/3d-tof-cameras/terabee-3dcam/

10. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

11. https://towardsdatascience.com/fooling-facial-detection-with-fashion-d668ed919eb

12. https://github.com/chanddu/Face-Recognition

13. https://docs.opencv.org/4.x/dc/d88/tutorial_traincascade.html

14. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

15. https://realpython.com/python-sockets/

16. https://github.com/OpenKinect/libfreenect