به نام خدا

گزارش نهایی

آز سخت‌افزار - گروه ۳- دکتر اجلالی

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم‌سال دوم ۰۰-۰۱

نویسندگان:

علی حاتمی تاجیک-۹۸۱۰۱۳۸۵

امیرمحمد عیسی‌زاده-۹۸۱۰۶۸۰۷

*محمدحسین قیصریه-۹۷۱۰۶۲۳۸

چکیده

با گسترش تجهیزات کامپیوتری در سراسر جهان و افزایش کارایی و عملکرد آنها، زندگی بشر در ابعاد مختلف دچار تغییر و بهبود شده است. هوشمند سازی تجهیزات پیرامون از یک طرف و از طرف دیگر سادگی کار با تجهیزات زمینه جذاب و پرکابردی است که اخیرا بسیار مورد توجه قرار گرفته است. در واقع مسئله مورد نظر برگرفته از یک مشکل واقعی که انجام راحت تر دستورات کاربر با کامپیوتر است. در این پروژه ما سعی کردیم که هوشمند سازی ماوس را انجام دهیم و سامانه ای طراحی کنیم که بر اساس حرکات دست، دستورات کاربر را اجرا کند. علاوه بر اجرای دستورات عادی، به کاربر این امکان داده میشود که دستورات خاص تری را با حرکات تعریف شده دست انجام دهد. [۱]

کلیدواژه ها: کنترل تصویری، ماوس تصویری، پردازش تصویر

[۱] کدها، گزارشات و باقی فایل‌های مورد نیاز از طریق مخزن گیت‌هاب گروه که از طریق این لینک قابل بارگیری است.

فهرست مطالب

۱ مقدمه

در این فصل به طور خلاصه در مورد انگیزه اصلی پروژه و اهمیتش در زندگی روزمره صحبت خواهد شد.

۱.۱ تعریف مسئله

پردازش تصویر و فهمیدن فرمان انسان با پردازش حرکات آن از جانب کامپیوتر در دهه های اخیر مورد توجه بسیاری قرار گرفته است. اینکار میتواند فواید گوناگونی داشته باشد. اول اینکه با هوشمند سازی فهم دستورات، میتوانیم در وقت صرفه جویی کنیم و تجربه راحت تر و لذت بخش تری را برای کاربر ایجاد کنیم. در واقع سختی کار با ماوس فیزیکی، مسئله ای اجتناب ناپذیر است. بنابراین با حذف این آپشن، ماوس های تصویری جایگزین میشوند. این کار به کمک پردازش تصویر گرفته شده از دست کاربر که مکان و جابجایی های آن به حرکت نشانگر موشواره و عملیات های مربوط به آن منجر می شود. در کنار این تعدادی حالات قابل تعیین برای کاربر برای روانتر شدن داشته رابط کاربری وجود داشته باشد. در این پروژه نیازی به ماژول شدت صوت نیست و تمام حالت ها به وسیله پردازش تصویر قابل دستیابی هستند.

۲.۱ اهمیت مسئله

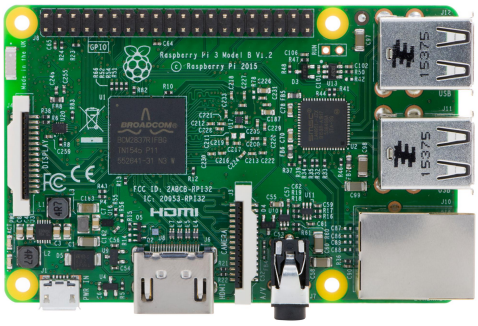
این نکته شناخته شده است که کارکرد آسان تر وسایل جانبی می تواند علاوه بر تجربه مناسب کاربر، به دلیل راحتی استفاده، اثرات مثبت ذهنی و روانی نیز روی کاربر بگذارد. بدین جهت انسان از مدت ها قبل اقدام به استفاده از وسایل آسان تر به مرور زمان(مثل صفحات لمسی و ...) کرد؛ در دنیای امروزی ممکن است به دلیل کارهای زیاد و همزمان، رسیدگی به آنها برای کاربر دشوار باشد. بنابراین هر روشی که بتواند برای دادن راحت تر فرمان به رایانه به کاربر کمک بکند، اهمیت بالایی دارد. به طور خاص این پروژه می تواند به انجام دستورات توسط دست و با استفاده از پردازش تصویر کمک کند. استفاده زیاد از وسایل جانبی مانند ماوس، در درازمدت میتواند آسیب های فیزیکی به انسان بزند که مشکلات فراوانی از قبیل گرفتگی مچ دست، اذیت شدن انگشتان و ... را در پی دارد.

از سوی دیگر اگر به بعد های صنعتی بپردازیم، متوجه خواهیم بود که مسئله کمبود راهکار مناسب برای ارتباط با رایانه، مسئله بسیار جدی در شرکت های الکترونیکی است.

نکته دارای اهمیت این است که سال های زیادی است که شرکت های بزرگ به سمت ایده های جدید برای کار با وسایل الکترونیکی رفته اند، به عنوان مثال لمسی کردن صفحات گوشی ها و امروزه لمسی کردن صفحات بعضی از لپ تاپ ها، ولی مسئله این است که لمسی کردن لپتاپ ها از جهت هایی، کاری است که فلسفه درستی ندارد، چون قدرت پردازش و برق و ... برای جایی دارد مصرف میشود که نیاز کاربر نیست، به عبارتی کاربر اگر میخواست که فرمان هایش با لمس مانیتور اتفاق بیافتد که تلفن های همراه نیاز او را برطرف می‌کردند، نیاز اصلی کاربر این است که بتواند با ابزار های ساده کننده ای کار کند که حس استفاده از رایانه و نه تلفن همراه را به او القا کند. علاوه بر تمام دلایلی که آوردیم، لمس مانیتور کاری سخت است، چون ابعاد صفحه بزرگ است که موجب خستگی کاربر میشود.

۳.۱ راهکار پیشنهادی

در این پروژه هدف داریم راهکاری برای دادن فرمان های کاربر به رایانه با صرف انرژی کم ارائه دهیم، همچنین این امکان را برای کاربر فراهم سازیم تا با راحت تر شدن کار، زمان کمتری را صرف کند. در این قسمت راه حل پیشنهادی را به طور کلی مطرح می کنیم. توجه داشته باشید به طور کلی هرچقدر ما شناخت دقیق تری از نیاز کاربر داشته باشیم، آسان سازی ارتباط با رایانه به نحوه مطلوب تری صورت می گیرد. ما ابزاری را طراحی کردیم که کاربر با صرف انرژی بسیار کم، در همان جایی که هست، با استفاده از نوک انگشتانش، از راه دور دستوراتش را به رایانه بدهد. راه حل پیشنهادی ما بدین صورت است که با قراردادن یک دوربین در کنار نمایشگر، از حرکات کاربر فیلم برداری شود. سپس برای کم کردن بار پردازشی از روی پردازنده رایانه، از یک برد Raspberry Pi 3B+ استفاده شده است که تصاویر گرفته شده توسط دوربین، به آن داده می‌شود. سپس تحلیل ها برروی پردازش گر جانبی انجام میشود، و ساختار دست کاربر و حرکات آن بررسی می‌شود. در نهایت تحلیل های انجام شده در قالب دستوراتی به رایانه فرستاده می‌شود. این کار باعث قدرت استقلال



شکل ۱: برد رزبری‌پای

از سیستم عامل می‌شود که بسیار بهتر از حالتی است که نیاز داریم پردازش ها بر روی رایانه انجام گیرد. در نهایت پردازش های انجام شده بر روی پردازنده جانبی، با استفاده از کابل سریال، به رایانه فرستاده می‌شود و رایانه آن دستورات را تبدیل به دستورات IO می‌کند.

۴.۱ ساختار گزارش

دراین گزارش سعی شده است که ابتدا نیاز مندی های سخت افزاری بررسی شود. بنابراین درفصل بعدی ابتدا به بررسی ماژول های سخت افزاری و انتظاری که از آنها داریم پرداخته می شود. سپس در فصل سوم به بررسی مشخصات ماژول های سخت افزاری انتخابی می پردازیم. در ادامه وارد نکات پیاده سازی پروژه می شویم. در فصل چهارم هم به آنالیز کد نرم افزاری پیاده سازی شده می پردازیم. در فصل آخر هم یک جمع بندی کلی پروژه داریم و مسائلی که در ادامه باید به آنها فکر شود، می پردازیم.

۲ سخت افزار مورد نیاز

۱.۲ واحد پردازشی

برای واحد پردازشی از برد Raspberry Pi 3B+ استفاده شده است. [۱] (شکل ۱)

Raspberry Pi 3 مجهز به یک پردازنده چهار هسته ای 64 بیتی Broadcom BCM2837 ARM Cortex-A53 SoC است که با فرکانس ۲.۱ گیگاهرتز کار می کند، که آن را حدود ۵۰ درصد قدرتمندتر از Pi 2 می کند. این بدان معناست که Raspberry Pi 3 جدید می تواند برای برنامه های آفیس و مرور وب.

نوآوری بزرگ در این نسخه سوم بدون شک اضافه شدن تراشه WiFi و بلوتوث کم انرژی است. این نه تنها باعث صرفه جویی در فضا می شود (دیگر نیازی به اتصال دانگل های WiFi و بلوتوث ندارید)، بلکه پورت های USB بیشتری را برای اتصال دستگاه های دیگر آزاد می کند.

با افزودن این دو ویژگی، Raspberry Pi روشن کرده است که این نسخه جدید برای اینترنت اشیا (IoT) و اتوماسیون خانگی طراحی شده است. Raspberry Pi 3 همچنین با Windows 10 IoT Coreسازگار است، سیستم عاملی که برای ایجاد و توسعه برنامه های کاربردی برای اتوماسیون خانگی، روباتیک و اشیاء متصل طراحی شده است.

برد Raspberry Pi 3 به اندازه Raspberry Pi 2 است و کانکتور و پیکربندی اجزا تقریباً یکسانی دارد. تنها چیزی که تغییر کرده است موقعیت LED ها است که به سمت دیگر کارت SD منتقل شده اند تا فضایی برای آنتن WiFi ایجاد

شکل ۲: دوربین OVA5647

کنند. همه کانکتورها در یک مکان قرار دارند و عملکردهای یکسانی دارند. بنابراین می توان از لوازم جانبی Pi 2 و B+ خود با RasPi 3 نیز استفاده کرد.
مشخصه‌های سخت‌افزاری این برد به صورت زیر است:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

۲.۲ تصویربرداری

برای بخش تصویربرداری که به نوعی آشیل این پروژه به حساب می‌آید از ماژول دوربین رسپبری پای (5MP, 1080p, v1.3) استفاده شده است. (شکل ۲)
دوربین برد رسپبری پای با استفاده از کانکتور CSI مستقیما به برد رسپبری پای متصل می شود. این دوربین توانایی تصویر برداری با وضوح ۵ مگاپیکسل و ضبط ویدیو با کیفیت 1080p HD را دارد. این ماژول با طراحی سفارشی خود توسط بنیاد رسپبری پای در انگلستان ساخته شده و دارای سنسوری با فوکوس ثابت و رزولوشن 1944×2592 است. این

شکل ۳: ماژول TTL-USB

ماژول از طریق کابل ریبون ۱۵ پینه به رسپبری پای متصل می شود و از رابط سریال دوربین که مختص اتصال دوربین ها ساخته شده ، استفاده می کند. CSI قابلیت انتقال نرخ بسیار بالایی از داده ها را دارد.

برد این ماژول کوچک و به اندازه ی 20x25x9 میلیمتر و وزن آن تنها ۳ گرم است، به همین دلیل برای استفاده در پروژه هایی که وزن و ابعاد قطعات اهمیت بالایی دارند، گزینه ی بسیار مناسبی است. سنسور این ماژول دارای رزولوشن ۵ مگاپیکسل و لنز آن دارای فوکوس ثابت است که آن را قادر به دریافت و ارائه ی تصاویر استاتیک با کیفیت 2592x1944 پیکسل و همچنین قابلیت پشتیبانی از 60fps @ 720p ,30fps @ 1080p و 60/90 640x480p برای ضبط ویدیو می سازد.

این ماژول تا آخرین نسخه ی Raspbian که سیستم عامل مورد استفاده در رسپبری پای است را پشتیبانی می کند.
مشخصه‌های این دوربین به شرح زیر است:

- سازگاری کامل با مدل ها A و B رسپبری پای
- ماژول دوربین 5MP Omnivision 5647
- رزولوشن عکس 2592x1944
- پشتیبانی از 30fps @ 1080p، 60fps @ 720p و 60/90 640x480p برای ضبط ویدیو
- رابط سریال دوربین-۱۵ پین MIPI - تغذیه مستقیم از برد رسپبری پای
- ابعاد : 9 x 25 x 20 میلیمتر
- وزن : ۳ گرم

۳.۲ ارتباطات

برای ارتباط بین برد رسپبری‌پای و رایانه مقصد از ماژول سریال TTL استفاده شده است که به صورت کابل درامده است و استفاده راحت‌تری و شکیل‌تری نسبت به ماژول خام دارد. (شکل ۳)
مشخصات فنی این کابل:

- بر اساس ماژول PL2303HX
- شاخصه سیم‌ها نیز به صورت زیر است:
  - قرمز : پاور

* سیاه : زمین
* سفید : RX در پورت USB
* سبز : TX در پورت USB

۴.۲ بسته‌بندی محصول

برای بسته‌بندی این محصول برای استفاده حداکثری از فضا و زیبایی و شکیل بودن محصول نهایی تصمیم گرفته شده است تا بسته‌بندی محصول توسط پرینترهای سه‌بعدی تهیه شوند. این شکل بسته‌بندی به این دلیل که انعطاف‌پذیری زیادی برای طراح دارد می‌تواند طرح‌های زیبا و چشم‌نوازی را خلق کند و باعث افزایش فروش محصول به خاطر ظاهر زیبا شود. البته این روش (پرینتر سه‌بعدی) برای تعداد بالای محصول پاسخگو نیست و در صورتی که تولید انبوه صورت بگیرد با استفاده از قالب، و پالیش نهایی می‌توان با سرعت بالاتری به همان نتیجه رسید با این هزینه که تهیه تجهیزات مورد نیاز برای این کار بسیار بالاتر خواهد بود. در اصل پرینترهای سه‌بعدی اکثرا برای طرح اولیه استفاده می‌شوند. شکل ۴ نشان‌دهنده طرح‌های اولیه برای بسته‌بندی است. البته این طرح‌ها با احتمال خوبی دستخوش تغییر خواهد شد.

۳ نحوه پیکربندی سخت‌افزار

۱.۳ دیاگرام سیستم

در شکل ۵ دیاگرام کلی طرز کار سیستم آمده است. روند کلی کار به این صورت است که ابتدا تصویر از محیط اطراف گرفته [۲] می‌شود. سپس با استفاده از رابط CSI به واحد پردازشی منتقل می‌شود. زمانی که تصویر در واحد پردازشی دریافت شد مدل هوش مصنوعی که از قبل تمرین‌داده [۳] شده است روی تصویر پردازش انجام می‌دهد که نتیجه آن تشخیص محل دست‌ها و نقاط عطف آن است. پس از آن، این نقاط عطف تفسیر می‌شوند و نتیجه با استفاده از رابط سریال به رایانه مقصد منتقل می‌شود و از آنجا با استفاده از درایورش تفسیر نهایی و تبدیل به حرکات موشواره روی آن صورت می‌گیرد.

۲.۳ پیکربندی

پیکربندی نرم‌افزاری شامل دو مورد است. اول اتصال دوربین به برد که با استفاده از یک کابل صاف [۴] انجام می‌گیرد. (شکل ۶)

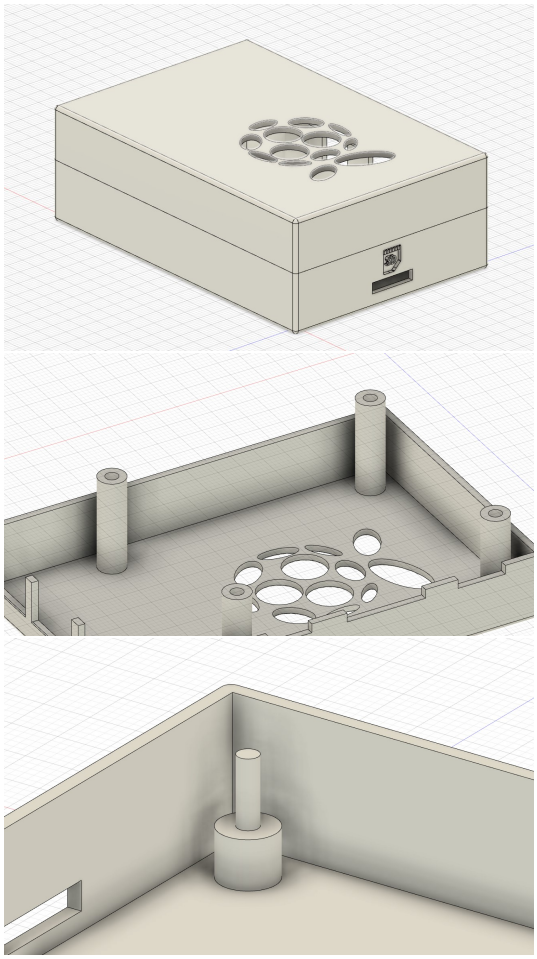
پیکربندی ماژول TTL نیز به این صورت انجام میگیرد که پایه‌های TX, RX به صورت برعکس به مشابهشان در رزبری وصل می‌شوند و همچنین پایه زمین [۵] به زمین برد متصل می‌شود تا زمین‌ها مشترک باشند. البته بنا بر این بود تا سیم قدرت نیز به برد متصل شود و قدرت مورد نیاز برد از طریق درگاه USB لپ‌تاپ تامین شود اما با توجه به معیوب بودن ماژول خریداری شده و انتقال قدرت نامناسب و ناامن و غیرقابل اعتماد [۶] تصمیمم بر این شد که توان مورد نیاز برد به صورت مستقیم از طریق منبع توان تامین شود. پیکربندی این ماژول در شکل ۷ قابل مشاهده است.

۴ نحوه پیاده‌سازی نرم افزاری

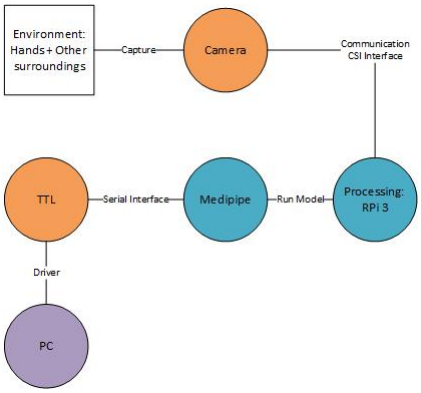
در ابتدا از تصمیم بر استفاده از ماژول های نوشته شده و آماده بود ولی بعد از تست کردن برخی از آن‌ها و کارکرد نامطلوبشان، تصمیم گرفتیم که این ماژول ها را خودمان طراحی کنیم.

---

[۲] Capture
[۳] train
[۴] flat
[۵] GND, Ground
[۶] unreliable

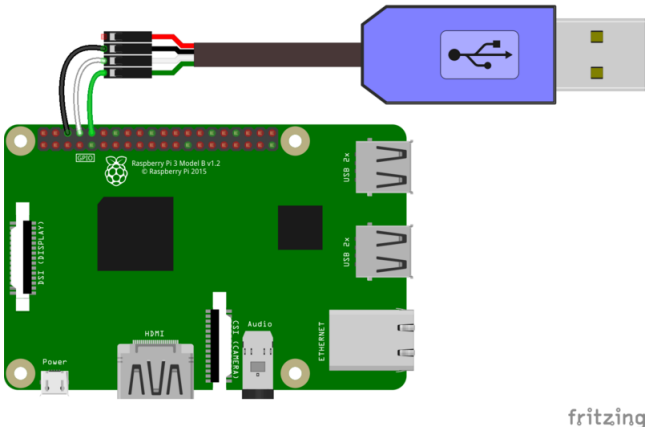


شکل ۴: بسته‌بندی محصول



شکل ۵: دیاگرام سیستم

شکل ۶: اتصال دوربین رزبری‌پای



شکل ۷: اتصال ماژول سریال

کد های پروژه شامل سه بخش اصلی است که هر کدام به تفصیل توضیح داده خواهند شد. برای پیاده سازی نرم افزاری پروژه نیاز داریم تا در ابتدا ماژولی دست ما را تشخیص دهد، سپس حرکات دست تعبیر شوند، سپس این تعابیر به صورت دستوری خلاصه مانند کلیک راست، یا حرکت نشانگر یا ... درآورده شوند. سپس این تعابیر از پردازنده جانبی به رایانه فرستاده شوند، سپس با توجه به نوع دستور فرستاده شده، تعبیر IO معلوم شود و در نهایت این تعبیر به صورت یک دستور IO انجام شود.

تمامی کد های این پروژه با استفاده از زبان برنامه نویسی python ورژن ۳ زده شده است. بخش تشخیص دست و انگشتان و ... در فایل `handdetector.py` آورده شده است. بخش تعبیر حرکات و تبدیل یک سری حرکات مشخص به دستور خاص در فایل `handler.py` آورده شده است.

۱.۴ ماژول handdetector.py

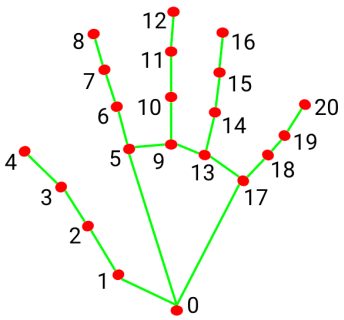
این ماژول که بخش اصلی و هسته این پروژه است وظیفه تشخیص دست، پیدا کردن مکان نقاط از پیش تعیین شده، پیدا کردن فاصله بین دو نقطه و پیدا کردن اینکه کدام انگشت ها در حال حاضر بالا هستند را دارد. نقطه اصلی این ماژول، کتابخانه mediapipe است که کار های پردازش تصویر را انجام می‌دهد. این کتابخانه توسط گوگل توسعه یافته. برای این منظور یک کلاس HandDetector تعریف شده و این توابع در آن پیاده سازی شده‌اند. کتابخانه mediapipe امکانات فراوانی از جمله تشخیص دست، تشخیص چهره، تشخیص محدوده مو و ... را دارد. کارکرد این کتابخانه در این پروژه قسمت تشخیص دست آن یا همان hand detection است. استفاده ما از این کتابخانه در این پروژه پیدا کردن مکان یک سری نقاط مشخص روی دست است و سایر کار ها را از روی مکان این نقاط انجام می‌دهیم.

شکل ۸ همان ۲۰ نقطه مد نظر است، می‌توان گفت که مهم ترین بین این نقاط، نقطه هشت یا همان نوک انگشت اشاره است. این نقطه معلوم کننده جای نشانگر ماوس است. در ادامه می‌بینید که مثلا از فاصله این نقطه با دست، می‌توان فهمید که انگشت اشاره بالا است یا خیر. [۲]

برای نشان دادن محدوده دست و جای انگشتان و ... بر روی نمایشگر از کتابخانه openCV استفاده می‌شود. تصاویر ضبط شده با کتابخانه mediapipe توسط این کتابخانه نمایش داده می‌شوند. [۳]

هم‌اکنون به توابع این فایل می‌پردازیم و کارکرد آنها را با هم بررسی می‌کنیم.

```
1 def findHands(self, img, draw=False):
```



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

شکل ۸: نقاط عطف (Landmark)

```
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imgRGB)

    if draw:
        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                self.mp_drawing.draw_landmarks(img, handLms,
                                               self.mp_hands.HAND_CONNECTIONS)
```

تابع findHands: این تابع برای پیدا کردن دست(ها) استفاده می‌شود، به طوری تصویر گرفته شده از وبکم را دریافت می‌کند سپس آن را به صورت یک عکس cv2 در می‌آورد و در نهایت این عکس را به تابع mediapipe.hands.process می‌دهد، که نتیجه آن آبجکت های دست ها است که در متغیر hands ذخیره می‌شوند و اگر مقدار متغیر draw برابر با True باشد، جای دست و ... را روی تصویر می‌کشد.

```
def findPosition(self, img, handNo=0, draw=False):
    xList = []
    yList = []
    bbox = []
    self.lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(myHand.landmark):
            # print(id, lm)
            h, w, c = img.shape
            cx, cy = int(lm.x * w), int(lm.y * h)
            xList.append(cx)
            yList.append(cy)
            # print(id, cx, cy)
            self.lmList.append([id, cx, cy, lm.z])
            if draw:
                cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

        xmin, xmax = min(xList), max(xList)
        ymin, ymax = min(yList), max(yList)
        bbox = xmin, ymin, xmax, ymax

```

```
        if draw:
            cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax +
    20),
                          (0, 255, 0), 2)

    return self.lmList, bbox
```

این تابع برای ذخیره سازی جایگاه نقطه های مورد نظر است. به این شکل که این نقاط را در متغیر های XList و YList که لیست هستند می‌ریزد. و در نهایت اگر متغیر draw برابر با True باشد دور دست یک مستطیل می‌کشد یا به عبارتی محدوده آن را معلوم می‌کند. محدوده دست به این شکل است که پایین ترین و بالاترین xها و y ها را ذخیره می‌کند و از هر ظرف ۲۰ پیکسل حاشیه می‌دهد تا به خوبی معلوم شود.

```
def findDistance(self, p1, p2, img, draw=False, r=15, t=3):
    x1, y1 = self.lmList[p1][1:3]
    x2, y2 = self.lmList[p2][1:3]
    cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

    if draw:
        cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
        cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
        cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
        cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
    length = math.hypot(x2 - x1, y2 - y1)

    return length, img, [x1, y1, x2, y2, cx, cy]
```

این تابع فاصله بین دو نقطه عطف را در دست پیدا می‌کند (فاصله اقلیدسی) و اگر متغیر draw مقدار یک داشته باشد یک خط بین دو انگشت می‌کشد و بین آنرا با رنگ قرمز پر می‌کند که نشان‌دهنده این باشد که فاصله بین این دو انگشت اندازه گرفته شده است.

۲.۴ ماژول handler.py

این ماژول که مسئول تعبیر حالت های دست و اطلاعات دریافت شده از ماژول handdetector.py است، شامل یک حلقه while بی‌نهایت است که در هر لحظه اطلاعات را تفسیر می‌کند و آنها را به دستو قابل فهم تبدیل می‌کند و آن را به صورت سریال برای رایانه ارسال می‌کند.

این ماژول به صورت یک سرویس روی برد رزبری‌پای پیکربندی شده است تا با هربار بوت شدن سیستم (اتصال به برق) به صورت خودکار اجرا شده و کارهای مربوط به خود را انجام دهد.

کتابخانه های اصلی این ماژول، کتابخانه های numpy و serial و ماژول handdetector هستند. یکی از اتفاقات مهمی که در این ماژول رخ می‌دهد، ست کردن اندازه وبکم است، چون اندازه ای که دوربین می‌گیرد، 2592x1944 پیکسل است که پردازش این مقدار پیکسل برای مدل هوش مصنوعی بسیار زمان بر است، پس اندازه تصویر را در ابتدا به 320x240 پیکسل تغییر می‌دهیم. بقیه قسمت های این ماژول را برای تفهیم بهتر کارکرد آنها با شکل نمایش می‌دهیم.

در حلقه در هر لحظه تصویر خوانده می‌شود و سپس جای دست و انگشت ها به خصوص انگشت اشاره و انگشت وسط پیدا می‌شوند. سپس در تابع fingersUp که در ماژول handdetector پیاده سازی شد، انگشت هایی که بالا هستند شناسایی می‌شوند. شایان به ذکر است که متغیر fingers یک لیست پنج تایی است که نمایانگر پنج تا انگشت است، اگر مقدار انگشتی یک بود یعنی آن انگشت بالا و اگر مقدارش صفر بود یعنی پایین است. حال با توجه به متغیر هایی که تا الان تعریف کردیم، چندین تفسیر برای عکس وجود دارد که به شرح زیر است:

- حالت حرکت: این حالت زمانی رخ می‌دهد که فقط انگشت اشاره بالا باشد و انگشت وسط پایین باشد، در این

صورت به حالت حرکت می‌رویم و در این حالت هر جا که نوک انگشت اشاره برود، به نقطه ای در صفحه نمایشگر نگاشت می‌شود.

```
    # 4. Only Index Finger : Moving Mode
        if fingers[1] == 1 and fingers[2] == 0:
          # 7. Move Mouse
          ser.write (b"v*%d*%d\n"%(x1,y1))
```

- حالت کلیک دوتایی: این حالت زمانی است که هر دو انگشت اشاره و وسط بالا باشند، به این حالت می‌رویم. به این صورت که وقتی هر دو بالا می‌روند کار دیگری نمی‌توانیم بکنیم و به عبارتی نمی‌توانیم نشانگر را تکان دهیم، تا زمانی که یا از این حالت بیرون بیاییم یا اینکه دو تا انگشت را به هم نزدیک کنیم و اگر فاصله از ۱۵ پیکسل کمتر شد، دابل کلیک انجام می‌شود.

```
if fingers == [0,1,1,0,0]:
    print('dclick-mode')
    # 9. Find distance between fingers
    length, img, lineInfo = detector.findDistance(8, 12, img)
    # 10. Click mouse if distance short
    if length < 15:
        if not dclicked:
            ser.write(b"d\n")
            dclicked = True
    else:
        dclicked = False
```

- حالت کلیک تکی: این حالت زمانی است که انگشت اشاره و دو انگشت وسط بالا باشند، سپس اگر فاصله انگشت اشاره و انگشت وسط کمتر از ۱۵ پیکسل شد، چپ کلیک اتفاق می‌افتد و مقدار clicked هم True می‌شود و در غیر این صورت مقدار آن False می‌شود.

- حالت درگ‌اندراپ: وقتی انگشت اشاره و انگشت شست با هم بالا باشند به حالت drag and drop می‌رویم، به این صورت که اگر فاصله این دو انگشت از ۱۵ پیکسل کمتر شود mousedown یا همان drag رخ می‌دهد و اگر در حالت drag باشیم و فاصله بیشتر از ۱۵ پیکسل شود، mouseup یا همان drop می‌کنیم و مقدار mouseDown هم برابر با False میگذاریم.

- حالت راست کلیک: این حالت زمانی اتفاق می‌افتد که انگشت اشاره و انگشت شست و انگشت وسط با هم بالا باشند. اگر فاصله انگشت اشاره و انگشت وسط کمتر از ۱۵ پیکسل شود، راست کلیک اتفاق می‌افتد.

- حالت اسکرول: این حالت زمانی اتفاق می‌افتد که تمامی انگشت ها بجز شست بالا باشند به حالت اسکرول می‌رویم. حال اگر دستمان را بالا رویم، اسکرول پایین اتفاق می‌افتد و اگر دستمان را پایین ببریم، اسکرول بالا اتفاق می‌افتد(مثل صفحه تاچ لپ‌تاپ که وقتی به بالا دستمان را می‌بریم، صفحه پایین می‌آید و وقتی پایین می‌بریم، بالا می‌رود.)

```
if fingers == [0,1,1,1,1]:
    # scroll mode
    pos = y1
    if last_pos_scroll == -1:
        last_pos_scroll = pos
    else:
        if abs(last_pos_scroll - pos) > 10:
```

```
            clicks = int((last_pos_scroll - pos))
            ser.write (b"s*%d\n"%(clicks))
            last_pos_scroll = pos
```

- اسکرین‌شات: با باز کردن انگشت کوچک و انگشت شصت و بستن باقی انگشتان این حالت فعال می‌شود. برای جلوگیری از اسکرین شات گرفتن مداوم فاصله زمانی یک ثانیه‌ای بین هر دستور اسکرین شات قرار گرفته است.

```
if fingers == [1,0,0,0,1]:
    if time.time_ns() - last_ss > 1_000_000_000:
        last_ss = time.time_ns()
        ser.write (b"p\n")
```

۳.۴ درایور

درایور آن بخشی از از قسمت نرم‌افزاری است که در سمت رایانه مقصد اجرا می‌شود. در موضوعی که ما از آن استفاده می‌کنیم درایور ابتدا به رابط سریال محصول متصل می‌شود. سپس ارتباطی را با آن آغاز کرده و اگر ارتباط امکان پذیر باشد دستورات را از آن می‌خواند و عملیات‌های لازم را به سیستم‌عامل اعلام می کند.
درایور مورد استفاده ما از دو فرآیند [۷] کلی تشکیل شده است:

- Interpreter: این بخش مربوط به گرفتن و خواندن دستورات و اعمال تغییرات لازم بر سیستم‌عامل است. در این قسمت پس از اینکه دستورات از سریال که توسط ویژوالایزر انتخاب شده است (توضیح در بعد) یک خط از دستورات را خواند ابتدا چک می‌کند که فرمت ارسالی دستور درست باشد و داده‌های اشتباهی فرستاده نشده‌باشد. سپس اگر فرمت دستور با فرمت دستوراتی که برای برنامه تعریف شده است هم‌خوانی داشته باشد، با توجه به نوع دستور (اولین بایتی که ارسال می‌شود) به صورت داینامیک تابع مربوط به آن انتخاب می‌شود و پارامترهایی (یک یا چند عدد صحیح) که با آن ارسال شده است به تابع پاس داده می‌شود تا عملیات‌های خود را انجام دهد.

  عملیات‌های انجام شده توسط کتابخانه `pyautogui` انجام می‌شود که امکان فرستادن دستورات موشواره و کیبورد را به سیستم‌عامل دارد البته برای اینکه بتوان از این کلیک‌ها و ... روی قسمت‌های امنتیتی ویندوز نیز استفاده کرد باید درایور به صورت Administrator اجرا شود. [۵]

خلاصه کدی که در این بخش اجرا می‌شود به شرح زیر است:

```
def get_command():
    # wait till a new command is received
    command = get_serial_line()
    if command:
        for pattern in patterns:
            # search in the patterns if the command exists or not
            x = re.search(pattern, command)
            if x:
                # seperate command and its parameters
                x = list(x.groups())
                return x[0], x[1:]
        return None, None
    else:
        # sleep to prevent busy waiting until a serial is connected
        time.sleep(0.5)
        return None, None
```

[۷] Process

```
17
18
19 def interpret(command: str, params: list):
20     # finds corresponding handler of the command and do the command
21     globals()["do_"+command](params)
22
23 def start():
24     global visualizer
25     visualizer = Visualizer()
26     visualizer.start()
27     while True:
28         try:
29             # get command from serial
30             command, params = get_command()
31             if command:
32                 # interprate command
33                 interpret(command, params)
34         except Exception as e:
35             # log bad habits
36             logger.log(3, str(e))
```

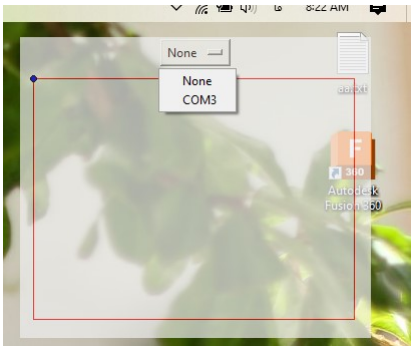
تابع `interpret` به این صورت کار می‌کند که از سیمبل‌هایی که به صورت جهانی در اسکریپت تعریف شده است آن سمبلی که با `do_` آغاز می‌شود و پایانش با نام کانمند ما است (توابع متناسب در فایل تعریف شده‌اند) را انتخاب و پاس دادن پارامترها اجرا کن.

لیست دستوراتی که درایور از آن پشتیبانی می‌کند نیز به صورت زیر است:

```
1  # patterns send from RPi
2  patterns = [
3      "^(b)[*]([-]?\d+)[*]([-]?\d+)[*]([-]?\d+)[*]([-]?\d+)$",
4      "^(v)[*]([-]?\d+)[*]([-]?\d+)$",
5      "^(d)$",
6      "^(md)$",
7      "^(mu)$",
8      "^(c)$",
9      "^(r)$",
10     "^(s)[*]([-]?\d+)$",
11     "^p$"
12 ]
```

- Visualizer: این بخش مسئول نشان‌دادن تفاسیر برنامه از حرکات کاربر به صورت گرافیکی و فراهم کردن ساز و کاری برای انتخاب پورت دستگاه است که البته در حال حاضر ساختار گرافیکی ساده‌ای دارد. به خاطر طویل بودن کدهای مربوط به GUI/ از آوردن آنها خودداری شده است اما کدها به صورت کامل کامنت‌گذاری شده‌اند. تصویری از این رابط کاربری در شکل ۹ آمده است. این رابط کاربری در زمان اجرا در گوشه بالا سمت راست است و برای اینکه کاربر بتواند تمام محتوای صحفه را ببیند به صورت محو در صفحه ظاهر می‌شود که البته توسط کاربر قابل جابجایی است و با درگ اند دراپ در هر جای صفحه می‌تواند قرار بگیرد. این نمایشگر همیشه روی برنامه‌های دیگر قرار می‌گیرد و زیر صفحه دیگری نمی‌رود. برای بستن آن نیز کافی‌است تا با راست کلیک روی آن از منوی مربوط گزینه Exit انتخاب شود.

  دارای یک کادر قرمزرنگ است که نشان‌دهنده صفحه کاربر (با اندازه دوربین) است. مستطیل سیاه (که در شکل مشخص نیست) نشان‌دهنده حدود دست است و دایره آبی رنگ نشان‌دهنده نوک انگشت سبابه کاربر است. در



شکل ۹: نمایی از ویژوالایزر درایور

بالاترین بخش آن نیز یک منوی کشویی [۸] قرار دارد که در هربار باز کردن این منو لیستی از پورت‌های سریال متصل به رایانه به کاربر نمایش داده می‌شود که کاربر باید پورت مربوط به محصول را از بین آن انتخاب کند.

۵ پیکربندی سیستم

در این بخش به پیکربندی‌هایی لازم برای سیستم (هم برد و هم رایانه مقصد) می‌پردازیم.

۱.۵ پیکربندی رسبپری‌پای

۱.۱.۵ نصب سیستم‌عامل ۶۴بیتی

برای اینکه بتوانیم از کتابخانه مدیاپایپ بر روی بردهای رزبری پای استفاده کنیم باید این برد به صورت ویژه پیکربندی بشود چراکه این کتابخانه برای سیستم‌عامل‌های ۳۲ بیتی توسعه نیافته است و در صورت نیاز برای استفاده از آنها کاربر باید خود این کتابخانه را بسازد[۹].

برای ساختن این کتابخانه به ابزار Bazel نیاز است که برای نصب آن روی بردهای رزبری‌پای به حداقل ۱۶ گیگابایت حافظه نیاز است که با توجه به اینکه حافظه فعلی ما یک SD ۸ گیگابایتی است این کار مقدور نبود و پس از چندین بار تلاش، به نتیجه‌ای نرسید.

برای همین باید یک سیستم‌عامل ۶۴ بیتی (که خوشبختانه چند ماهی است برای بردهای رزبری‌پای توسعه یافته است) استفاده کنیم. برای این کار باید تصویر [۱۰] مربوط به این سیستم‌عامل روی SD کارت نوشته شود که بهترین گزینه برای انجام این کار نرم‌افزار رسمی رسپبری‌پای یعنی Raspberry Pi Imager است. پس از انتخاب نسخه ۶۴ بیتی و درایو مورد نظر، این نرم‌افزار سیستم‌عامل را از منابع معتبر دانلود کرده و آن را روی کارت حافظه تصویر می‌کند (شکل ۱۰).

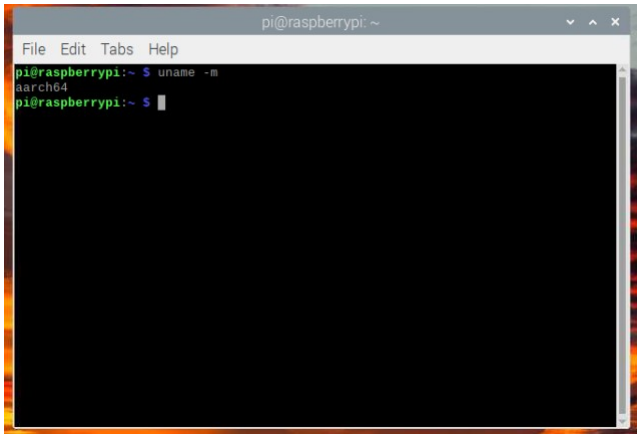
همچنین این نرم‌افزار کانفیگ‌های لازم برای راه‌اندازی برد به صورت Headless را نیز در اختیار ما قرار می‌دهد. این کار با استفاده از پیکربندی اتصال وای‌فای و فعال‌سازی سرویس SSH امکان‌پذیر است.

شکل ۱۱ نشان‌دهنده موفقیت‌آمیز بودن این عملیات است.

---

[۸] Drop Down Menu
[۹] Build
[۱۰] Image - iso file

شکل ۱۰: Raspberry Pi Imager



شکل ۱۱: سیستم‌عامل ۶۴ بیتی

۲.۵ نصب کتابخانه‌های لازم

پس از اینکه سیستم‌عامل به خوبی پیکربندی شد، باید کتابخانه‌هایی که سرویس تشخیص حرکات دست به آن نیاز دارد را نصب کرد. این کتابخانه‌ها به وسیله دستور `pip` قابل نصب هستند:

- NumPy
- OpenCV
- OpenCV-ControlLib
- ffmpeg
- MediaPipe
- MediaPipe-Hands-Solution
- PySerial

۱.۲.۵ راه‌اندازی سرویس تشخیص حرکات دست

ما نمی‌تونیم همانند زمانی که به رفع باگ پروژه مشغولیم با استفاده از سرویس SSH اسکریپت پایتون را راه‌اندازی کنیم. به همین دلیل باید این کار را به صورت یک سرویس در بیاوریم تا هر بار در ابتدای روش شدن رزبری به صورت خودکار به آن متصل شود.
برای این کار فایل `/etc/rc.local` را به شکل زیر آپدیت می‌کنیم:

```
sudo -E python ~/scripts/handler.py &
exit 0
```

با این کار اسکریپت مورد نظر در زمان بالاآمدن سیستم‌عامل به صورت دسترسی ادمین و با دسترسی به محیط [۱۱] کاربر اصلی (که روی آن کتابخانه‌ها نصب شده اند) اجرا می‌شود. با قرمز ممتد شدن چراغ دوربین متوجه می‌شویم که این سرویس به صورت کامل آغاز شده است و از آن به بعد قابل استفاده و در حال فرستادن دستورات به رابط سریال است.

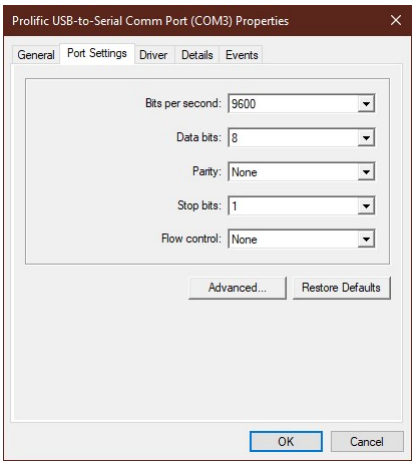
۳.۵ پیکربندی رایانه مقصد

علاوه بر کارهایی که روی برد رسپری‌پای باید صورت بگیرد، رایانه‌های مقصد نیز برای ارتباط قابل اطمینان و پایدار باید پیکربندی شوند.

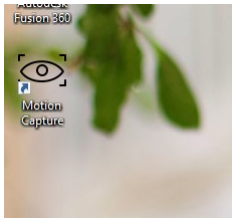
۱.۳.۵ پیکربندی رابط سریال

اولین گام برای پیکربندی رابط سریال روی سیستم مقصد نصب درایور ماژول سریال به شماره PL2303HX [۴] است. نصب این درایور برای برقراری ارتباط سریال الزامی است که البته می‌تواند به صورت پک نرم‌افزاری همراه با درایور اصلی برنامه به مشتری عرضه شود. این درایور باعث می‌شود تا ماژول و داده‌های ارسالی آن در رایانه مقصد به درستی تشخیص داده شود.
پس از نصب این درایور، رابط [۱۲] سریال باید پیکربندی شود تا تنظیمات یکسانی با برد داشته باشد تا این ارتباط برقرار باشد. برد برای برقراری امن از نرخ انتقال [۱۳] ۹۶۰۰ استفاده می‌کند. اطلاعات به صورت یک بایتی (هشت بیت) ارسال

[۱۱] Enviroment
[۱۲] Interface
[۱۳] Baud Rate



شکل ۱۲: پیکرندی رابط سریال در ویندوز



شکل ۱۳: فایل اجرایی درایور روی ویندوز

می شوند. بیتی که نشان‌دهنده پایان ارتباط است ۱ است و همچنین بیت موازنه [۱۴] نیز ارسال نمی‌شود. تنظیمات سریال مربوطه در ویندوز در شکل ۱۲ آمده است.

۲.۳.۵ پیکربندی درایور

درایور برای اینکه به درستی کار کند و در تمام محیط‌ها قابل استفاده باشد باید به صورت Administrator اجرا شود. برای این کار ابتدا یک فایل اجرایی می‌سازیم که درون آن با استفاده از کامندها درایور را که یک اسکرپیت پایتو است اجرا می‌کند. سپس یک میانبر [۱۵] برای آن می‌سازیم و آیکون آن را به چیزی ملموس تغییر میدهیم تا در دسکتاپ کاربر نمای خوبی داشته باشد. (شکل ۱۳)

۶ جمع‌بندی

در این بخش به مرور پروژه و برخی نکات باقی مانده می‌پردازیم.

[۱۴] Parity Bit
[۱۵] shortcut

۱.۶ مرور کلی

همانطور که پیش تر گفته شد، هدف اصلی بررسی ماژول های سخت افزاری مختلف و انتخاب آن ها در وهله اول بود. این کار سعی شد، با انجام جستجو هایی صورت بگیرد. در کل ماژول اصلی که مرتبط با کارمان بود، ماژول پردازنده بود که نوع Raspberry Pi 3 مورد استفاده قرار گرفت. انتخاب اول قطعا Raspberry Pi 4 و یا بردهای مشابه ۶۴بیتی و با قدرت پردازشی بالاتر بود، ولی به علت عدم دسترسی به این بورد، به ناچار نتوانستیم استفاده کنیم که این کار ما را با مشکل سرعت کم پردازش مواجه کرد. برای بهتر شدن پروژه تصمیم گرفتیم که یک ویژگی جدید اضافه کنیم که بجای استفاده از وای فای، از کابل برای انتقال داده استفاده کنیم. برای اینکار نیز کابل های مختلفی بود، اما برآن شدیم که از کابل TTL برای انتقال سریال داده استفاده کنیم که دارای سرعت مناسبی است. در ادامه برای نصب کتاب‌خانه mediapipe بر روی پردازنده، مجبور به حذف سیستم عامل ۳۲ بیتی و نصب سیستم عامل ۶۴ بیتی شدیم که همین کار، به علت سنگین بودن سیستم عامل ۶۴ بیتی برای پردازنده جانبی ما، باعث کم شدن سرعت شد. برای تصویر برداری هم میتوانستیم علاوه بر دوربین معمولی، از مادون قرمز هم استفاده کنیم که در شب هم نیاز کاربر را برطرف کند، ولی با کمبود امکانات از جمله قدرت پردازش پایین مواجه شدیم و از آن صرف نظر کردیم. برای اتصالات فیزیکی هم چون محصول دارای دو قطعه به اضافه کابل است، با مشکل خاصی مواجه نبودیم. در مرحله خرید ماژول ها آنها را تست کردیم تا مشکی نداشته باشند و دقت و عملکرد مناسبی داشته باشند. برای تامین برق پردازنده جانبی، ابتدا سعی کردیم که آن را با درگاه USB رایانه تامین کنیم، که چون برق کافی نمی‌رساند، پردازنده جانبی اطلاعات غلط ارسال می‌کرد و به همین دلیل آن را مستقیما به پریز برق وصل کردیم. برای ارسال داده همانطور که گفته شد دستورات تعبیر شده از حرکت دست از طریق ارتباط سریال به رایانه فرستاده می‌شود و در رایانه این دستورات، تبدیل به دستور IO می‌شود. برای پیاده سازی بخش نرم افزار تماما از زبان برنامه‌نویسی پایتون استفاده شد و می‌توان بخش نرم افزاری را به سه قسمت دسته بندی کرد:

۱. تشخیص دهنده دست: شاید اصلی ترین بخش پروژه همین قسمت باشد، چون پردازش تصویر و تشخیص حرکات در این بخش انجام می‌شود. این بخش در فایل `handdetector.py` پیاده سازی شد، وظیفه این بخش این است که تشخیص دهد دست کاربر الان در چه وضعیتی است. برای این کار با استفاده از کتابخانه mediapipe که توسط شرکت گوگل توسعه داده شده است، ۲۰ نقطه از نقاط دست را شناسایی می‌کند و با استفاده از این نقاط وضعیت دست های ما را می‌سنجد، یعنی می‌توان فهمید که کدام دست یا دست ها بالا هستند، فاصله انگشت ها نسبت به هم، تشخیص نوک انگشت اشاره و ...

۲. تبدیل کننده حرکات دست به فرمان: در فایل `handler.py` از ماژول تشخیص دست استفاده شده است که در یک حلقه بی‌نهایت ابتدا دست تشخیص داده می‌شود. سپس با توجه به باز یا بسته بودن انگشتان حالت عملیاتی از بین ۶ حالت مختلف تشخیص داده می‌شود و سپس با توجه به اینکه سبابه کجا قرار دارد، فاصله انگشتان چقدر است و ... این حرکات به دستورات ورودی و خروجی تفسیر شده و دستور مناسب از طریق رابط سریال همراه با پارامترهای مورد نیاز ارسال می‌شود.

۳. بخش تبدیل دستورات به فرمان های IO رایانه: دستورات تفسیر شده و ارسال شده از برد در این بخش یعنی درایور تبدیل به حرکات موشواره شده و به سیستم‌عامل منتقل می‌شوند. همچنین این ماژول در ریسه‌ای [۱۶] جدا تفسیرهای انجام گرفته را برای راحتی کاربر به نمایش می‌گذارد. همچنین این ریسه مسئول ساخت و پیکربندی پورت سریال است که این پورت به وسیله کاربر از ویژوالایزر انتخاب می‌شود.

۲.۶ هزینه های صورت گرفته

همانطور که مشخص است پروژه نیاز به ماژول ها و هزینه هایی داشت. اولا در قسمت نرم افزاری استفاده مان از پلتفرم mediapipe بود که رایگان بود، پس در بخش نرم‌افزاری نیازی به هزینه نبود. در مورد پردازنده که Raspberry Pi 3 بود، نیاز به تهیه نداشتیم و خوشبختانه توسط دانشگاه خریداری شده بود. به طور کلی هزینه هایمان که کابل سریال و

[۱۶] Thread

ردیف	پارت‌نامبر	هزینه (ریال)
۱	Raspberry Pi 3B+	۳۲،۰۰۰،۰۰۰
۲	Raspberry Pi Camera Module OVA5647	۱،۰۰۰،۰۰۰
۳	3D Printed Case	۳،۵۰۰،۰۰۰
۴	TTL to USB Cable PL2303HX	۵۰۰،۰۰۰

جدول ۱: هزینه‌ها

ماژول دارای دوربین است در جدول ۱ آمده اند. ممکن است برخی قیمت ها هم اکنون گران تر باشند که مسئله اجتناب ناپذیری است.

۳.۶ کارهای آینده

قطعا این پروژه یک مثال کوچک از یک پروژه صنعتی است که طبعا با آن ابعاد قابل قیاس نیست. اما به نظرمان همین پروژه کوچی هم می‌تواند گسترش های مناسبی داشته باشد. در ادامه سعی کردیم که برخی از ویژگی ها که می‌توان به پروژه اضافه کرد را بررسی کنیم.

- اضافه کردن سنسور های سخت افزاری دیگر: در این راستا می‌توانیم سنسور هایی نظیر مادون قرمز را داشته باشیم، که می‌تواند کارکرد دوبین را در شب هم انجام پذیر کند.
- استفاده از پردازنده‌های قدرتمندتر و بهره‌مندی از ممووری RAM
- تغییرات کد پیاده سازی شده: مطلب قابل تأمل این است که این طراحی برای افراد راست‌دست طراحی شده است، که با اضافه کردن امکانات بیشتر می‌تواند برای چپ‌دست ها هم پیاده سازی شود، که از دو طریق می‌توان انجام داد. مورد اول این است که فرد در ابتدا اعلام کند که چپ‌دست است یا راست‌دست و با توجه به آن، تحلیل های پردازنده انجام شود. حالت دیگر این است که پردازنده به صورت اتوماتیک با هر دو دست کار کند که خود این ویژگی شاید تجربه خیلی خوبی باشد، ولی نیاز به پیاده سازی های خیلی زیاد و هندل کردن مشکلاتی از جمله وقتی که هر دو دست بالا می‌روند دارد.
- اضافه کردن امکانات به سامانه: ویژگی های مختلفی را می‌توان به سامانه اضافه کرد. یک ویژگی که مدنظرمان بود این است که تایمری را داشته باشیم و هرگاه که پس از مدتی از سیستم استفاده نشد، برد پردازنده جانبی به حالت sleep برود تا در مصرف برق صرفه‌جویی شود.
- تغییرات رابط کاربری: برای داشتن رابط کاربری خود پلتفرم ویژگی های مختلفی را ارائه می‌دهد که برخی از آنها نیاز به هزینه داشت. مثلا یک ویژگی جالب برای رابطه کاربری این بود که کاربران بتوانند اپلیکیشنی را دانلود کنند و تمامی دستورات از جمله کلیک ها و ... را خودشان به دلخواه تایین کنند، که این کار نیاز به بررسی دقیق کتاب‌خانه mediapipe و تغییر قابل ملاحظه آن برای انعطاف پذیری بیشتر و حتی برای عملکرد خیلی خوب، نیاز به آموزش مدل یادگیری ماشین جداگانه است که نیاز به داده اولیه دارد.
- مسئله دوربین : دوربین فعلی نسبت به سایر دوربین ها از قیمت کمتری برخوردار است، ولی دو تا مشکل دارد، اولی تعداد فریم بر ثانیه است که گاها پایین است و برای حرکت های سریع دست کاربر شاید مشکل ساز شود. مورد دوم هم گستردگی زیادی ندارد یا به عبارتی گسترده [۱۷] تصویر برداری نمیکند که این مشکل می‌تواند با استفاده از دوربین ها بهتر حل شود. ولی راه حل ابتکاری می‌تواند استفاده از دو یا چند دوربین به صورت همزمان

[۱۷] Wide

باشد که در این صورت هم مشکل گسترده نبودن تصاویر حل می‌شود و علاوه بر آن مشکل بزرگ‌تری که تعداد فریم باشد هم تا حدود خوبی برطرف می‌شود.

مراجع

[۱] صفحه رسمی رزبری‌پای ۳ - [link]

[۲] کتاب‌خانه مدیاپایپ از گوگل - [link]

[۳] کتاب‌خانه پردازش تصویر OpenCV - [link]

[۴] صفحه رسمی ماژول سریال و درایور آن - [link]

[۵] رفع مشکل عدم کلیک روی کیبورد مجازی در ویندوز - [link]