

به نام خدا



گزارش نهایی

آز سخت افزار - گروه ۳ - دکتر اجلالی

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۰۱-۰۰

نویسندگان:

علی حاتمی تاجیک-۹۸۱۰۱۳۸۵

امیر محمد عیسی زاده-۹۸۱۰۶۸۰۷

*محمد حسین 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 پربابری است که اخیراً 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. کلیدواژه ها: کنترل تصویری، ماوس تصویری، پردازش تصویر



فهرست مطالب

۳	۱	مقدمه
۳	۱.۱	تعریف مسئله
۳	۲.۱	اهمیت مسئله
۳	۳.۱	راهکار پیشنهادی
۳	۴.۱	ساختار گزارش
۴	۲	سخت افزار مورد نیاز
۴	۱.۲	واحد پردازشی
۵	۲.۲	تصویربرداری
۶	۳.۲	ارتباطات
۶	۴.۲	بسته بندی محصول
۶	۳	نحوه پیکربندی سخت افزار
۶	۱.۳	دیاگرام سخت افزاری
۶	۲.۳	پیکربندی
۶	۴	نحوه پیاده سازی فیزیکی
۶	۵	نحوه پیاده سازی نرم افزاری
۷	۱.۵	ماژول handdetector.py
۹	۲.۵	ماژول handler.py
۹	۶	جمع بندی
۹	۱.۶	مرور کلی
۹	۲.۶	هزینه های صورت گرفته
۹	۳.۶	کارهای آینده



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.

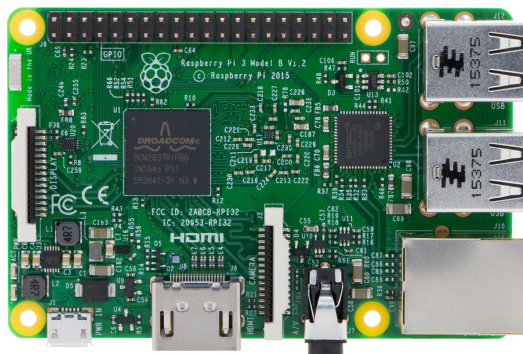
نکته دارای اهمیت این است که سال های زیادی است که شرکت های بزرگ به سمت ایده های جدید برای کار با وسایل الکترونیکی رفته اند، به عنوان مثال لمسی کردن صفحات گوشی ها و امروزه لمسی کردن صفحات بعضی از لپ تاپ ها، ولی مسئله این است که لمسی کردن لپ تاپ ها از جهت هایی، کاری است که فلسفه درستی ندارد، چون قدرت پردازش 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 می کردند، 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 می شود. سپس تحلیل ها بر روی پردازش گر جانبی انجام میشود، و ساختار دست کاربر و حرکات آن بررسی می شود. در نهایت تحلیل های انجام شده در قالب دستوراتی به رایانه فرستاده می شود. این کار باعث قدرت استقلال از سیستم عامل می شود 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 می شود و رایانه آن دستورات را تبدیل به دستورات IO می کند.

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.



شکل ۱: برد رزبری پای

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 Core سازگار است، سیستم عاملی که برای ایجاد و توسعه برنامه های کاربردی برای اتوماسیون خانگی، رباتیک 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 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.

مشخصه های سخت افزاری 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



شکل ۲: دوربین OVA5647

- 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 مستقیماً به برد رسپبری پای متصل می‌شود. این دوربین توانایی تصویربرداری 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 2592×1944 است. این ماژول از طریق کابل ریبون ۱۵ پینه به رسپبری پای متصل می‌شود و از رابط سریال دوربین که مختص اتصال دوربین‌ها ساخته شده، استفاده می‌کند. CSI قابلیت انتقال نرخ بسیار بالایی از داده‌ها را دارد. برد این ماژول کوچک و به اندازه 20x25x9 میلیمتر و وزن آن تنها ۳ گرم است، به همین دلیل برای استفاده در پروژه‌هایی که وزن و ابعاد قطعات اهمیت بالایی دارند، گزینه‌ی بسیار مناسبی است. سنسور این ماژول دارای رزولوشن ۵ مگاپیکسل و لنز آن دارای فوکوس ثابت است که آن را قادر به دریافت و ارائه‌ی تصاویر استاتیک با کیفیت 2592x1944 پیکسل و همچنین قابلیت پشتیبانی از 1080p @ 30fps, 720p @ 60fps و 640x480p 60/90 برای ضبط ویدیو می‌سازد.

این ماژول تا آخرین نسخه‌ی Raspbian که سیستم عامل مورد استفاده در رسپبری پای است را پشتیبانی می‌کند. مشخصه‌های این دوربین به شرح زیر است:

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



شکل ۳: ماژول TTL-USB

۳.۲ ارتباطات

برای ارتباط بین برد رسیبری پای و رایانه مقصد از ماژول سریال TTL استفاده شده است که به صورت کابل درآمده است و استفاده راحت تری و شکل تری نسبت به ماژول خام دارد. (شکل ۳) مشخصات فنی این کابل:

- بر اساس ماژول PL2303HX
- شاخصه سیم ها نیز به صورت زیر است:
 - * قرمز : پاور
 - * سیاه : زمین
 - * سفید : RX در پورت USB
 - * سبز : TX در پورت USB

۴.۲ بسته بندی محصول

برای بسته بندی این محصول برای استفاده حداکثری از فضا و

۳ نحوه پیکربندی سخت افزار

۱.۳ دیاگرام سخت افزاری

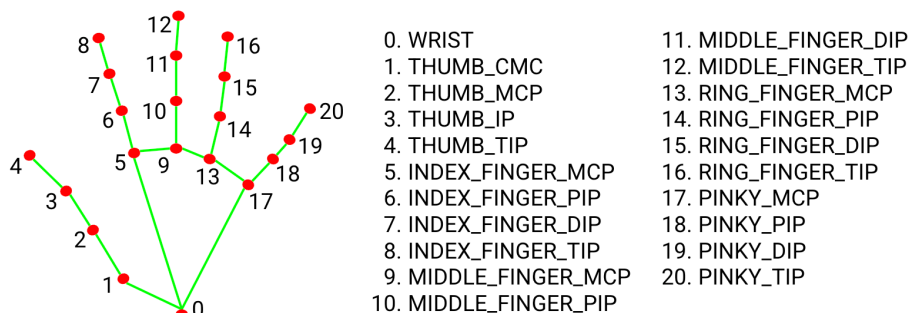
۲.۳ پیکربندی

۴ نحوه پیاده سازی فیزیکی

۵ نحوه پیاده سازی نرم افزاری

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

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 تعبیر به صورت دستوری خلاصه مانند کلیک راست، یا حرکت



شکل ۴: نقاط عطف (Landmark)

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

۱.۵ 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 کارهای 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 کارها را از روی مکان این نقاط انجام می‌دهیم.

شکل ۴ 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 مثلاً 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):
2     imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
3     self.results = self.hands.process(imgRGB)
4
5     if draw:
6         if self.results.multi_hand_landmarks:
7             for handLms in self.results.multi_hand_landmarks:
8                 self.mp_drawing.draw_landmarks(img, handLms,
9                                                  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 آبیجکت 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.

```

1 def findPosition(self, img, handNo=0, draw=False):

```




```
2     xList = []
3     yList = []
4     bbox = []
5     self.lmList = []
6     if self.results.multi_hand_landmarks:
7         myHand = self.results.multi_hand_landmarks[handNo]
8         for id, lm in enumerate(myHand.landmark):
9             # print(id, lm)
10            h, w, c = img.shape
11            cx, cy = int(lm.x * w), int(lm.y * h)
12            xList.append(cx)
13            yList.append(cy)
14            # print(id, cx, cy)
15            self.lmList.append([id, cx, cy, lm.z])
16            if draw:
17                cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
18
19            xmin, xmax = min(xList), max(xList)
20            ymin, ymax = min(yList), max(yList)
21            bbox = xmin, ymin, xmax, ymax
22
23            if draw:
24                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax
25                    + 20),
26                    (0, 255, 0), 2)
27
28        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 yها را ذخیره می‌کند و از هر ظرف ۲۰ پیکسل حاشیه می‌دهد تا به خوبی معلوم شود.

```
1 def findDistance(self, p1, p2, img, draw=False, r=15, t=3):
2     x1, y1 = self.lmList[p1][1:3]
3     x2, y2 = self.lmList[p2][1:3]
4     cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
5
6     if draw:
7         cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
8         cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
9         cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
10        cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
11        length = math.hypot(x2 - x1, y2 - y1)
12
13    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.



۲.۵ 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 دست‌قابل فهم تبدیل می‌کند و آن را به صورت سریال برای رایانه ارسال می‌کند. این ماژول به صورت یک سرویس روی برد رزبری پای 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.
- 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.
- 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 درگ‌اند‌دراپ: 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 دستان را بالا رویم، اسکرول پایین اتفاق می‌افتد و اگر دستان را پایین ببریم، اسکرول بالا اتفاق می‌افتد (مثل صفحه تاج لپ‌تاپ که وقتی به بالا دستان را می‌بریم، صفحه پایین می‌آید و وقتی پایین می‌بریم، بالا می‌رود).

۶ جمع‌بندی

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

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 ۶۴ بیتی و با قدرت پردازشی بالاتر بود، ولی به علت عدم دسترسی به این برد، 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



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

۱. چون محصول دارای دو قطعه به اضافه کابل است، با مشکل خاصی مواجه نبودیم. در مرحله خرید ماژول‌ها آنها را تست کردیم تا مشکلی 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 مستقیماً 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 تماماً 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 دست‌های ما را می‌سنجد، یعنی می‌توان فهمید که کدام دست یا دست‌ها بالا هستند، فاصله انگشت‌ها نسبت به هم، تشخیص نوک انگشت اشاره و ...

۲. تبدیل کننده حرکات دست به فرمان:

۳. بخش تبدیل دستورات به فرمان‌های IO رایانه:

۲.۶ هزینه های صورت گرفته

همانطور که مشخص است پروژه نیاز به ماژول‌ها و هزینه هایی داشت. اولاً 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 ماژول دارای دوربین است در جدول ۱ آمده اند. ممکن است برخی قیمت‌ها 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 طبعاً با آن ابعاد قابل قیاس نیست. اما به نظرمان همین پروژه کوچکی هم می‌تواند گسترش‌های مناسبی داشته باشد. در ادامه سعی کردیم که برخی از ویژگی‌ها که می‌توان به پروژه اضافه کرد را بررسی کنیم.

- اضافه کردن سنسورهای سخت افزاری دیگر: در این راستا می‌توانیم سنسورهایی نظیر مادون قرمز را داشته باشیم، که می‌تواند کارکرد دوربین را در شب هم انجام پذیر کند.

- استفاده از پردازنده‌های قدرتمندتر و بهره‌مندی از مموری 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 چپ‌دست‌ها 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، تحلیل‌های 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 سازی‌های خیلی زیاد و هندل کردن مشکلاتی از جمله وقتی که هر دو دست بالا می‌روند دارد.

- اضافه کردن امکانات به سامانه: ویژگی‌های 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 ویژگی‌های مختلفی را ارائه می‌دهد که برخی از آنها نیاز به هزینه داشت. مثلاً یک ویژگی جالب برای رابطه کاربری این بود که کاربران بتوانند اپلیکیشنی را دانلود کنند و تمامی دستورات از جمله کلیک‌ها 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 دوربین‌ها از قیمت کمتری برخوردار است، ولی دو تا مشکل دارد، اولی تعداد فریم بر ثانیه است که گاهی پایین است و برای حرکت‌های سریع دست کاربر شاید مشکل ساز شود. مورد دوم هم گستردگی زیادی ندارد یا به عبارتی گسترده^۱ تصویر برداری نمیکند که این مشکل می‌تواند با استفاده از دوربین‌ها بهتر حل شود. ولی راه حل ابتکاری می‌تواند استفاده از دو یا چند دوربین به صورت همزمان 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 رزبری پای ۳ - [link]
- [۲] کتابخانه مدیاپایپ از گوگل - [link]
- [۳] کتابخانه پردازش تصویر OpenCV - [link]