



دانشگاه تهران  
پردیس فارابی  
دانشکده مهندسی  
گروه مهندسی کامپیوتر

# توسعه برنامه‌ی دسکتاپ برای یک سیستم بازشناسی فعالیت‌ها بر اساس حسگر ژيروسکوپ موبایل

نگارش:

امیرحسین براری

استاد راهنما:

دکتر کاظم فولادی

گزارش پروژه برای دریافت درجه‌ی کارشناسی  
در رشته‌ی مهندسی کامپیوتر

شهریور ۱۳۹۹



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



## چکیده

اندازه گیری و آنالیز حرکات افراد کاربردهای بسیاری دارد. حرکت را می توان با استفاده از تکنیک ها و حسگرهای مختلفی اندازه گیری کرد. یکی از ساده ترین و مقرون به صرفه ترین ابزارها، سنسورهای تعبیه شده درون تلفن های همراه می باشد که در جهان امروز، همگی از آن بهره می برند. در تلفن های همراه، سنسورهای زیادی همچون سنسورهای مجاورت، سنسورهای دما، سنسورهای شتاب سنج، سنسورهایژیروسکوپ و... وجود دارد. با استفاده از برخی از این سنسورها مانند شتاب سنج وژیروسکوپ، می توان حرکات افراد را تشخیص داد.

در این پروژه، ابتدا با کمک اپلیکیشن های موبایل و دسکتاپ، دیتاست موردنظر برای چهار حرکت دست بالا، پایین، چپ و راست توسط مچ دست، جمع آوری شده و با استخراج ۱۰ ویژگی و پیاده سازی الگوریتم ماشین بردار پشتیبان (Support Vector Machine) به اختصار SVM، به تشخیص حرکت افراد پرداخته ایم.

پس از انجام تمامی کارها، ما به درصد میزان درستی ۹۶.۵٪ رسیدیم که درصد چشم گیری است. نتیجه تشخیص حرکت فرد را نیز در اپلیکیشن دسکتاپ، به صورت انیمیشن نشان می دهیم.

**کلمات کلیدی:** بازشناسی فعالیت، یادگیری ماشین،ژیروسکوپ، شتاب سنج، SVM، برنامه دسکتاپ



## فهرست مطالب

فصل اول	۸
مقدمه	۸
فصل دوم	۱۰
مروری بر کارهای گذشته	۱۰
فصل سوم	۱۴
ساختار سیستم بازشناسی فعالیت ها	۱۴
یادگیری ماشین یا یادگیری عمیق؟	۱۵
بررسی الگوریتم های مرتبط	۱۶
کتابخانه های استفاده شده در قسمت یادگیری ماشین پروژه	۲۲
توضیح کدهای پایتون مربوط به الگوریتم SVM	۲۲
فصل چهارم	۲۷
پیاده سازی برنامه دسکتاپ	۲۷
ماژول های ضروری Qt	۲۷
با Qt چه نرم افزار های معروفی توسعه داده شده اند؟	۲۸
برنامه اول: دریافت اطلاعات حسگرهای شتاب سنج و ژيروسکوپ از برنامه موبایل	۲۹
برنامه دوم: برنامه تشخیص حرکت ورودی	۳۹
فصل پنجم	۴۸
نحوه جمع آوری مجموعه داده	۴۸
نتایج و تحلیل آنها	۵۰
فصل ششم	۵۱
خلاصه	۵۱
نقاط قوت پروژه	۵۱
نقاط ضعف پروژه	۵۱
نتیجه گیری	۵۲
کارهای آینده	۵۲
مراجع	۵۳
پیوست	۵۴
نحوه کار با برنامه	۵۴

## فصل اول

### مقدمه

در دنیای امروزی، تلفن های همراه همه گیر هستند و روز به روز پیشرفته تر می شوند. یکی از اجزای اصلی سازنده ی تلفن های همراه هوشمند، حسگرها هستند. حسگرهای GPS، شتاب سنج، ژيروسکوپ، مجاورت، نور و اثر انگشت، تعدادی از حسگرهایی هستند که امروزه در تمامی تلفن های همراه هوشمند تعبیه شده اند. با وجود این سنسورها، کارهای زیادی را می توان انجام داد که یکی از آنها، تشخیص حرکت افراد می باشد. همچنین روز به روز از تکنولوژی برای راحتی افراد استفاده می شود و از تشخیص حرکات نیز می توان برای راحت تر کردن تجربه افراد استفاده کرد. برای مثال، اپلیکیشن هایی که برای سلامتی افراد بوجود آمده اند، با کمک حسگرها حرکات افراد را تشخیص می دهند و آخر روز تعداد قدم هایی که فرد در طول روز برداشته است را نشان می دهند.

این فرایند، شامل چند بخش کلی می باشد:

- جمع آوری دیتاست: در بعضی موارد دیتاست های موجود ممکن است کافی باشد و نیاز به جمع آوری نباشد، اما اگر دیتاست مرتبط وجود نداشته باشد، قدم اول جمع آوری دیتاست می باشد.
- تقسیم بندی: شناسایی مهمترین قسمت های داده بدست آمده
- استخراج ویژگی: یافتن ویژگی های تاثیر گذار و مهم از روی رفتار داده ها
- کاهش ویژگی ها: اگر تعداد داده هایمان زیاد باشد و تعداد ویژگی های استخراج شده نیز زیاد باشد، باید بررسی شود که از بین این ویژگیها، کدامشان واقعا حیاتی هستند و آن تعداد را فقط نگه داریم.
- طبقه بندی: داده ها و ویژگیهایشان را به الگوریتم موردنظر می دهیم تا به یادگیری بپردازد.
- استدلال: تعیین کلاس فعالیت داده شده

برای پروژه انتخابی اینجانب، پس از بررسی و خواندن مقالات به این نتیجه رسیدم که با کمک ۲ حسگر شتاب سنج و ژيروسکوپ، می توان حرکات دست افراد را تشخیص داد. حسگر شتاب سنج مقدار جابجایی تلفن همراه را در امتداد محور  $X$ ،  $Y$  و  $Z$  برمی گرداند و حسگر ژيروسکوپ مقدار چرخش بدن را در امتداد محور  $X$  یعنی حرکت از یک طرف به طرف دیگر، محور  $Y$  یعنی چرخش به عقب و جلو و محور  $Z$  یعنی چرخش از حالت عمودی به افقی و بالعکس را بر می گرداند. مورد و چالش بعدی انتخاب ویژگی های مناسب برای استخراج و مورد آخر انتخاب الگوریتم مناسب برای یادگیری بود. پس از بررسی و اعمال الگوریتم های مختلف، به این نتیجه رسیدم که الگوریتم SVM بهترین گزینه می باشد.



روش کار کلی ما در این پروژه، به این صورت بود که ابتدا دیتاست را جمع آوری کردیم و پس از آن ویژگیهای مدنظرمان را استخراج نموده و داده ها را به الگوریتم SVM داده ایم. نتیجه کار نیز درصد درستی پیش بینی ۹۶.۵٪ بود که درصد قابل قبولی است.

## فصل دوم

### مروری بر کارهای گذشته

در سال های اخیر، با توجه به پیشرفته بسیار سریع هوش مصنوعی و یادگیری ماشین، به مبحث تشخیص فعالیت توجه ویژه ای شده و پژوهش های زیادی در این زمینه صورت گرفته است که با توجه به مقالات موجود، می توان آنها را به دو دسته کلی تقسیم کرد. دسته اول تشخیص حرکات به کمک حسگرها است و دسته دوم تشخیص حرکات به کمک تصاویر ویدئویی می باشد. در جدول زیر، مقالاتی که در این زمینه کار شده را بر اساس هدفشان نشان می دهد.

ردیف	دسته بندی	شماره مرجع	محتویات پژوهش
۱	نظارت و امنیت	[۱]	<ul style="list-style-type: none"> <li>فعالیت های مشکوک یا خشونت آمیز را از ویدئو تشخیص می دهد.</li> <li>در صورت تشخیص فعالیت های مشکوک، به فرد میزبان اخطار می دهد.</li> </ul>
۲	نظارت و امنیت	[۲]	<ul style="list-style-type: none"> <li>ناهنجاری ها را با کمک یادگیری عمیق در فیلم دوربین های مدار بسته تشخیص می دهد.</li> <li>مجموعه ای از دیتاست های واقعی را از دوربین های مدار بسته مختلف جمع آوری و تایید می کند.</li> </ul>
۳	سلامت	[۳]	<ul style="list-style-type: none"> <li>از سنسورهای پوشیدنی برای تشخیص فعالیت های پیچیده انسان استفاده می کند.</li> <li>حسگرهایی را درون محیط اتاق قرار می دهد تا فعالیت را دقیق تر تحلیل کند.</li> </ul>
۴	سلامت	[۴]	<ul style="list-style-type: none"> <li>از مهندسی ویژگی خودکار برای بهبود تشخیص فعالیت در برنامه AAL استفاده می کند.</li> <li>فعالیت ها را می تواند تنها با تلفن همراه هوشمند یا ساعت هوشمند و با دقت بالا تشخیص دهد.</li> </ul>
۵	رانندگی خودمختار	[۵]	<ul style="list-style-type: none"> <li>حرکات راننده را با بررسی وضعیت سر و چشم فرد، تشخیص میدهد.</li> <li>مجموعه ای جدید از ویژگیهای مبتنی بر سر و چشم را ارائه می دهد.</li> </ul>
۶	تعامل انسان - ربات	[۶]	<ul style="list-style-type: none"> <li>یک سیستم رباتیک نو برای پردازش سیگنال دارای چند سنسور تعریف می کند.</li> <li>نتایج امیدوارکننده را در تشخیص حرکات انسان نشان</li> </ul>

			می دهد.
۷	تعامل انسان - ربات	[۷]	<ul style="list-style-type: none"> <li>• مدل و الگوریتمی جدید را پیشنهاد می کند که می تواند به طور خودکار فیلم های ضبط شده توسط یک ربات را پردازش کند، در حالی که با مردم تعامل می کند.</li> <li>• در حین تعامل با انسان، مجموعه ای از احساسات انسان را تشخیص می دهد.</li> </ul>

حال اگر بخواهیم مقداری مفصل تر به کارهای مشابه از گذشته تا امروز بپردازیم، به موردهای زیر می توان اشاره کرد.

- یکی از تلاشهای اولیه برای شناسایی فعالیتهای با استفاده از حسگرها ، استفاده از حسگرهایی مانند شتاب سنج و ژيروسکوپ، به منظور شناسایی طیف وسیعی از فعالیت ها مانند دست دادن، بلند کردن تلفن، گذاشتن تلفن، باز کردن در، نوشیدن و استفاده از قاشق بود. بدین منظور، مجموعه ای از حسگرها بر روی بازوی داوطلبان قرار می دادند که یک سر آن بر روی مچ دست و یک سر دیگر آن در بالای بازو، نزدیک شانه قرار داشت. داده های جمع آوری شده شامل اطلاعات جهت گیری نسبی مانند زاویه نگه داشتن دست و حرکت دست بود و تفسیر نتایج با استفاده از این دو ویژگی صورت گرفت. مورد اول دارای درصد درستی ۸۷٪ بود و مورد دوم دارای درصد درستی ۶۲٪ بود.
- یک راه حل دیگر استفاده از شتاب سنج تلفن همراه برای شناسایی فعالیت های انسانی بود. در آن زمان، این تنها سنسور حرکتی قابل توجه بود که بیشتر دستگاه های تلفن همراه از آن استفاده می کردند. در این فاصله ، چندین حسگر دیگر به تلفن های همراه هوشمند اضافه شدند. فعالیت های تحت نظر شامل راه رفتن، دویدن آهسته، نشستن، ایستادن، بالا رفتن از پله ها و پایین آمدن از پله ها بود. این مطالعه به سه بخش اصلی تقسیم شد:

○ جمع آوری داده ها

○ تولید ویژگی ها

○ کارهای تجربی

اولین مرحله ، جمع آوری داده ها ، با کمک بیست و نه داوطلب انجام شد که هنگام انجام فعالیت ها یک دستگاه تلفن همراه را در جیب شلوار خود حمل می کردند. از آنجا که این راه حل مستلزم استفاده از الگوریتم طبقه بندی است، در مرحله بعد، هدف اصلی شکل دادن به داده ها بود تا بتوان آنها را به عنوان ورودی به الگوریتم های موجود منتقل کرد.

در این مرحله شش نوع ویژگی ایجاد شد:

○ میانگین

○ انحراف معیار

○ میانگین تفاضل قدر مطلق‌ها

○ شتاب متوسط نتیجه

○ زمان بین قله‌ها

○ توزیع باند

مرحله آخر، بخش عملی آزمایش با ویژگی‌های استخراج شده و با سه روش طبقه‌بندی، یعنی درخت تصمیم، رگرسیون لجستیک و شبکه‌های عصبی چند لایه است. نتایج نشان داد که درصد خوبی برای پیاده‌روی (۹۲٪)، آهسته‌دویدن (۹۸٪)، نشستن (۹۴٪)، و ایستادن (۹۱٪) تشخیص داده شد. اینها مقادیر متوسط سه روش طبقه‌بندی هستند. در حالی که بالا رفتن از پله و پایین آمدن به طور متوسط ۴۹٪ و ۳۷٪ بود. اگر رگرسیون لجستیک در نظر گرفته نمی‌شد، به ۶۰٪ و ۵۰٪ افزایش می‌یافت. این منجر به این شد که در نهایت این دو فعالیت به کلی حذف شوند. در عوض، یک فعالیت جدید برای بالا رفتن از پله‌ها با دقت متوسط ۷۷٪ اضافه شد، که هنوز هم به طور قابل توجهی کمتر از چهار فعالیت دیگر به دست آمده بود، اما قابل اطمینان‌تر است.

• یکی از جدیدترین و پیچیده‌ترین مطالعات، راه حل مبتنی بر تلفن‌های هوشمند و حسگرهای حرکتی مچ دست را پیشنهاد می‌کند. ایده اصلی این روش این است که نحوه نگهداری تلفن‌های هوشمند توسط کاربرانشان (به عنوان مثال در جیب شلوار) برای تشخیص فعالیت‌های انسانی که شامل حرکات دست است، مناسب نیست. به همین دلیل از سنسورهای اضافی به غیر از سنسورهای دستگاه استفاده می‌شود. هر دو مجموعه سنسور شامل شتاب سنج، ژيروسکوپ و شتاب خطی بودند.

داده‌های استفاده شده برای ۱۳ فعالیت از ۱۰ شرکت‌کننده جمع‌آوری شد، اما فقط ۷ فعالیت توسط همه شرکت‌کنندگان انجام شد که دقیقاً همان فعالیت‌هایی است که راه حل پیشنهادی با هدف شناسایی آنها انجام می‌شود. هر فعالیت به مدت ۳ دقیقه انجام شد، مقدار کل ۳۰ دقیقه برای هر فعالیت بود و در نتیجه یک مجموعه داده ۳۹۰ دقیقه‌ای ایجاد شد. تنها ۲ ویژگی میانگین و انحراف معیار استخراج شد. این ویژگی‌ها برای پیچیدگی کم و دقت معقول برای فعالیت‌های مختلف انتخاب شدند و نتایج از ترکیب حسگرها تشکیل شد. هر سنسور به تنهایی و سپس در ترکیب با حسگرهای دیگر و همچنین ترکیب دو موقعیت برای دستگاه‌های تلفن همراه ارزیابی شد. هنگام استفاده از شتاب سنج و ژيروسکوپ در حالت مچ دست، در تشخیص فعالیت‌ها چند خطا رخ داد و بزرگترین سردرگمی بین راه رفتن و بالا رفتن از پله، با دقت ۵۳٪ بود. نتایج هنگام ترکیب دو موقعیت آزمون بهبود یافت، دقت کلی به ۹۸٪ افزایش یافت. اشکال اصلی این راه حل، وقتی با راه حل پیشنهادی مقایسه می‌شود، استفاده از دو دستگاه تلفن همراه بود که در زندگی واقعی غیرممکن است.

- یکی از زمینه هایی که اخیراً بسیار مورد توجه قرار گرفته است فعالیت های ورزشی به ویژه تناسب اندام و دویدن است. نمونه های بی شماری از برنامه ها که از شناسایی فعالیت های انسانی برای کمک به کاربران در پیگیری جلسات تمرینی خود وجود دارند، مانند:

○ Samsung Health

○ Nike+

○ Mi Fit

اگر چه این برنامه ها تعداد بسیار محدودی از فعالیت ها را تشخیص می دهند، اما نتایج بسیار خوبی دارند و در تشخیص نوع فعالیت انجام شده بسیار دقیق هستند و به مشتریان خود مزایایی مانند مکث خودکار هنگام تشخیص اینکه کاربر دیگر در حال دویدن نیست و همچنین شخصی سازی برنامه تمرینی می دهند.

بعلاوه در سالهای اخیر، در ساعت های هوشمند و بندهای تناسب اندام رشد چشمگیری مشاهده شده است. مانند Fitbit و Apple Watch که می توانند تعداد قدم ها، الگوی خواب، دوره های بی تحرکی و... را ردیابی کنند. این نوع دستگاه ها به عنوان اجزای سیستم های پیچیده تری استفاده می شوند که حسگرهای زیادی را برای انجام فعالیت های عمیق برای سناریوهایی مانند مراقبت های بهداشتی، سلامت افراد پیر، فناوری اقناعی برای رفتار سالم و... را به کار می گیرند.

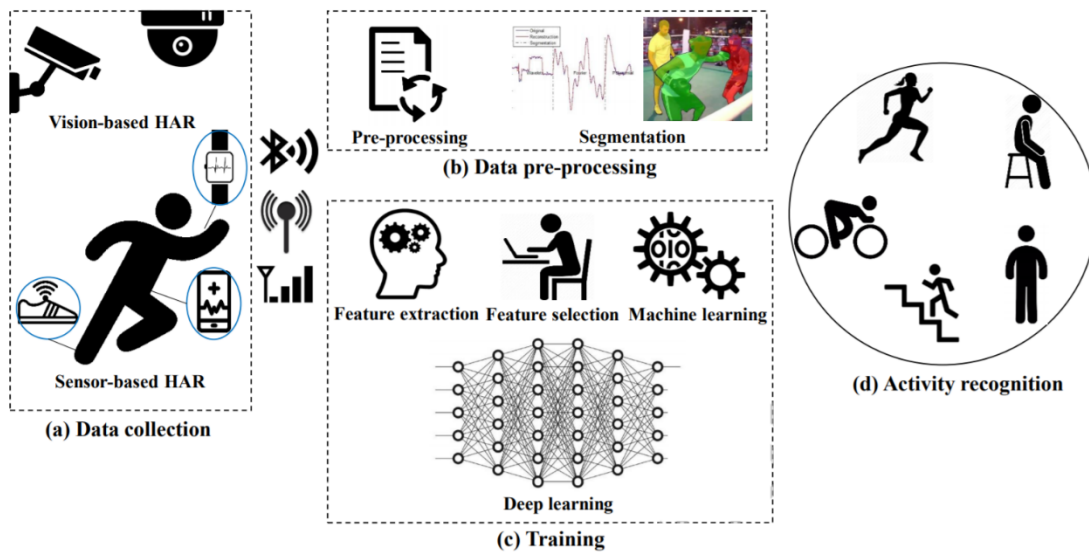
- لذت مردم از بازی هرگز از بین نمی رود و در زمینه بازی پیشرفت های زیادی حاصل شده است. در چند سال گذشته ، با ایجاد فناوری هایی مانند Kinect، PlayStation Move و Nintendo Wii ، شناسایی فعالیت به بخشی فعال در بازی تبدیل شده است. اگرچه برخی از این فعالیت ها را فقط با استفاده از محاسبات بصری تشخیص می دهند (مانند Kinect) ، سایر موارد به حسگرها متکی هستند. Nintendo Wii دارای یک ریموت است که دارای یک سنسور حرکتی است که شناسایی فعالیت را امکان پذیر می کند. همه اینها به روشهای مختلفی برای انجام بازی ها و حتی ایجاد عادت سالم مورد استفاده قرار می گیرند. با شناخت فعالیتهایی که کاربر در حال انجام آن است، رایانه می تواند کاربر را درک کرده و بر اساس واکنش های انسانی پاسخی ارائه دهد. از این طریق تعامل انسان و کامپیوتر امکان پذیر است.

- مطالعات اخیر نشان داده است که می توان از فعالیت و رفتار انسان برای نشان دادن وضعیت سلامتی انسان استفاده کرد. تحقیقات انجام شده، متخصصان پزشکی را به این نتیجه رسانده است که بین میزان فعالیت بدنی و بیماری های مختلف مربوط به چاقی و متابولیسم ارتباط زیادی وجود دارد. با توجه به مقدار زیادی از داده های قابل جمع آوری در مورد فعالیت فرد ، ایده استفاده از این داده ها برای جمع آوری اطلاعات در مورد وضعیت پزشکی انسان به سرعت رشد کرده است. اگرچه اعتقاد بر این است که این ممکن است راه حل بهتری نسبت به یک قرار ملاقات پزشکی با محدودیت زمانی باشد، اما به عنوان یک جایگزین در نظر گرفته نمی شود، بلکه بیشتر شبیه یک ابزار اضافی است.

## فصل سوم

### ساختار سیستم بازشناسی فعالیت ها

در این بخش، به تفصیل درباره نحوه کارها و الگوریتم ها می پردازیم. در ابتدا توسط تصویر زیر، مراحل کار بازشناسی فعالیت های انسان را نمایش می دهیم.



در تصویر بعدی، ۵ دسته بندی که مربوط به فعالیت ها می شود اشاره می شود. این دسته بندی ها عبارتند از حرکات، اعمال، تعامل با اشیاء، تعامل با انسانها و فعالیت گروهی.



## یادگیری ماشین یا یادگیری عمیق؟

سالهای زیادی است که از یادگیری ماشین در تشخیص فعالیت انسان استفاده می شود و نتایج قابل قبولی نیز ارائه کرده است اما نیاز است که ویژگی هایی که می خواهیم استخراج شوند را خودمان پیدا کنیم. از طرف دیگر در یادگیری عمیق، اگر به تعداد زیاد و قابل قبولی داده تهیه شود، الگوریتم، ویژگی های مناسب را پیدا کرده و استخراج می کند. اینکه کدام روش مناسبتر است بستگی به ابعاد پروژه دارد. در این پروژه تشخیص داده شد که از یادگیری ماشین استفاده شود.

بطور کلی مقایسه این ۲ روش در جدول زیر آورده شده است.

یادگیری عمیق	یادگیری ماشین	
<ul style="list-style-type: none"> <li>پیش پردازش و نرمال سازی برای بهبود عمل کرد الزامی نیست.</li> </ul>	<ul style="list-style-type: none"> <li>پیش پردازش و نرمال سازی برای بهبود عمل کرد الزامیست.</li> </ul>	پیش پردازش داده
<ul style="list-style-type: none"> <li>ویژگی های مناسب را خودش پیدا می کند.</li> </ul>	<ul style="list-style-type: none"> <li>خود فرد باید ویژگی مناسب را استخراج کند.</li> <li>در فعالیت های پیچیده احتمال شکست دارد.</li> <li>در تعداد داده زیاد، باید تعدادی ویژگی خاص انتخاب شوند.</li> </ul>	مهندسی ویژگی
<ul style="list-style-type: none"> <li>نیازمند دیتاست بزرگ است.</li> <li>پیچیدگی محاسباتی بالایی دارد.</li> <li>به سخت افزار قوی برای ترین کردن نیاز دارد.</li> </ul>	<ul style="list-style-type: none"> <li>با داده های مخصوص ترین کوچک به خوبی کار می کند.</li> <li>نیازمند زمان محاسبات و استفاده حافظه محدود است.</li> </ul>	فرآیند یادگیری

## Random Forest ( جنگل تصادفی )

یک الگوریتم یادگیری ماشین با قابلیت استفاده آسان است که اغلب اوقات نتایج بسیار خوبی را حتی بدون تنظیم فرایارامترهای آن، فراهم می کند. این الگوریتم به دلیل سادگی و قابلیت استفاده، هم برای طبقه بندی (Classification) و هم رگرسیون (Regression)، یکی از پرکاربردترین الگوریتم های یادگیری ماشین محسوب می شود.

Random Forest یک الگوریتم ترکیبی (ensemble) بوده که از درخت های تصمیم، برای الگوریتم های ساده و ضعیف خود استفاده می کند. در الگوریتم جنگل تصادفی از چندین درخت تصمیم (برای مثال ۱۰۰ درخت تصمیم) استفاده می شود. در واقع مجموعه ای از درخت های تصمیم، با هم یک جنگل را تولید می کنند و این جنگل می تواند تصمیم های بهتری را (نسبت به یک درخت) اتخاذ نماید.

در الگوریتم جنگل تصادفی به هر کدام از درخت ها، یک زیرمجموعه ای از داده ها تزریق می شود. برای مثال اگر مجموعه داده ای دارای ۱۰۰۰ سطر (۱۰۰۰ نمونه) و ۵۰ ستون (۵۰ ویژگی) بود، الگوریتم جنگل تصادفی به هر کدام از درخت ها، ۱۰۰ سطر و ۲۰ ستون، که به صورت تصادفی انتخاب شده اند و زیر مجموعه ای از مجموعه ی داده ها هست، می دهد. این درخت ها با همین دیتاست زیر مجموعه، می توانند تصمیم بگیرند و مدل طبقه بند خود را بسازند.

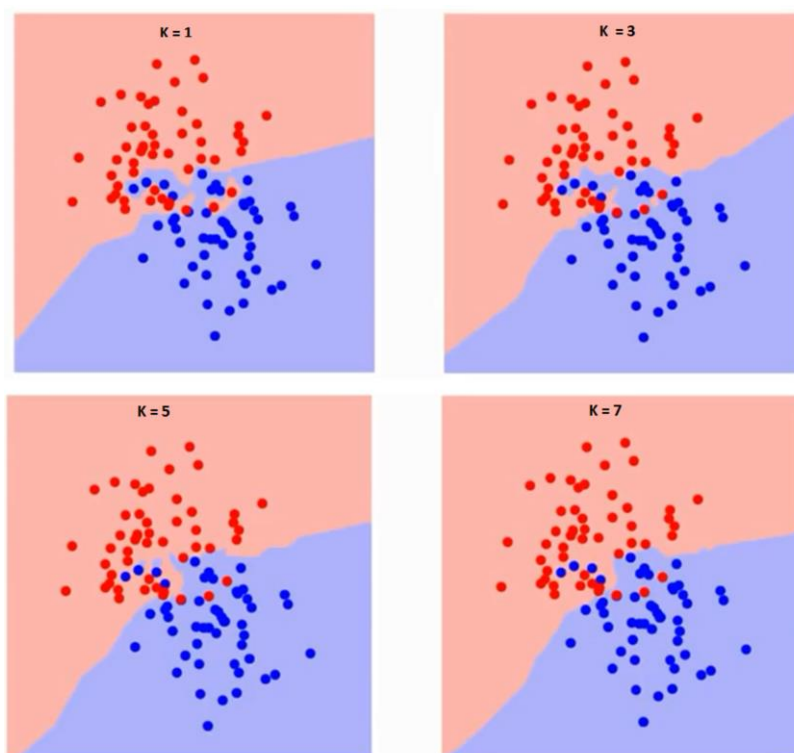
یکی از بزرگ ترین مشکلات در یادگیری ماشین، بیش برآزش (overfit) است، اما اغلب اوقات این مساله به آن آسانی که برای دسته بند جنگل تصادفی به وقوع می پیوندد، اتفاق نمی افتد. محدودیت اصلی جنگل تصادفی آن است که تعداد زیاد درخت ها می توانند الگوریتم را برای پیش بینی های جهان واقعی کند و غیر موثر کنند.

## KNN ( k-نزدیک ترین همسایگی )

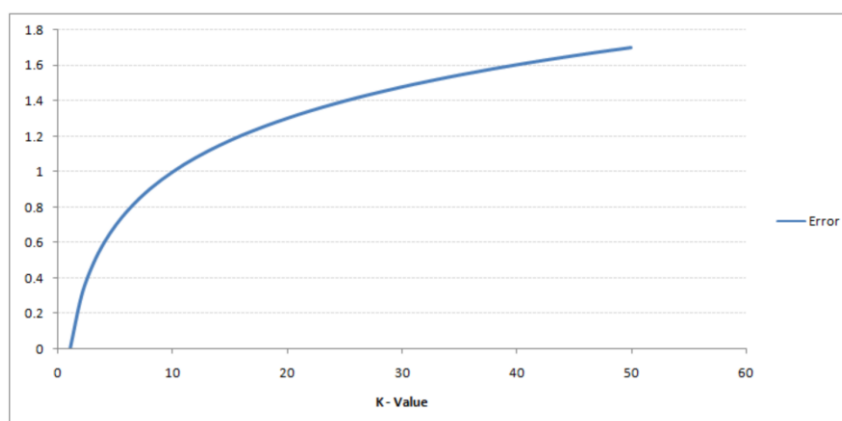
یک روش ناپارامتری است که در داده کاوی، یادگیری ماشین و تشخیص الگوریتم مورد استفاده قرار می گیرد. بر اساس آمارهای ارائه شده، الگوریتم k-نزدیک ترین همسایگی یکی از ده الگوریتمی است که بیشترین استفاده را در پروژه های گوناگون یادگیری ماشین و داده کاوی، هم در صنعت و هم در دانشگاه داشته است.



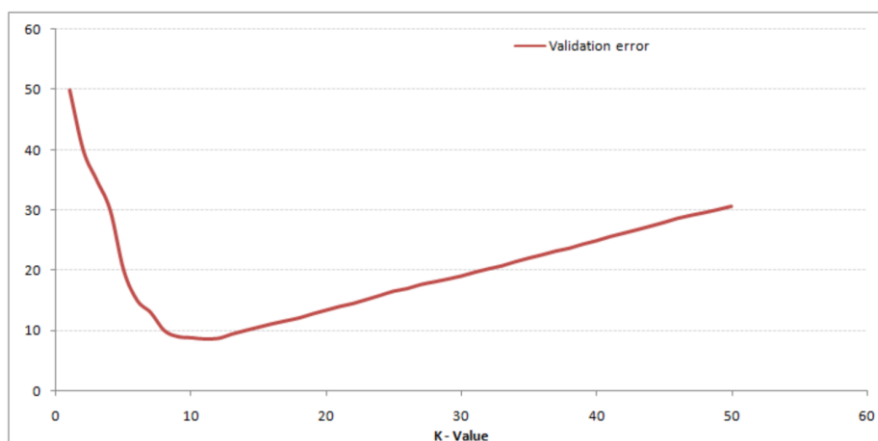
در ابتدا، تأثیر انتخاب  $k$  در الگوریتم  $k$ -نزدیک‌ترین همسایگی بر نتایج خروجی در تصویر زیر آمده است.



همان‌طور که از تصاویر مشهود است، با افزایش مقدار  $k$ ، مرزهای کلاس‌ها روان‌تر می‌شود. با افزایش  $k$  به سمت بی‌نهایت، بسته به اکثریت مطلق نمونه‌ها در نهایت یک کلاس آبی یا قرمز وجود خواهد داشت. نرخ خطاهای آموزش (training) و ارزیابی (evaluation) دو عاملی هستند که برای تعیین مقدار  $k$  مناسب به آن‌ها نیاز داریم. نمودار خطی در شکل زیر بر اساس مقدار خطای آموزش برای  $k$ های گوناگون ترسیم شده است.



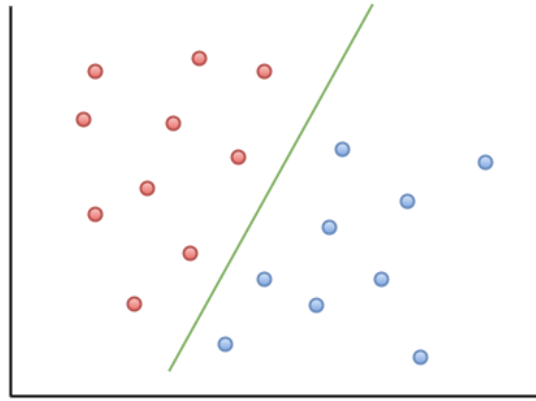
چنان که مشاهده می شود، نرخ خطا هنگامی که  $k=1$  است، برای نمونه های آموزش صفر باقی می ماند. این امر بدین دلیل است که نزدیک ترین نقطه به هر داده آموزش، خود آن داده است. بنابراین نتایج پیش بینی با  $k=1$  صحیح است. اگر  $k=1$  باشد نمودار خطای ارزیابی نیز روند مشابهی دارد. نمودار خطای ارزیابی با مقادیر گوناگون  $k$  در شکل زیر نمایش داده شده است.



هنگامی که  $k=1$  است مرزها دچار بیش برازش (overfit) شده اند. بنابراین نرخ خطا کاهش پیدا کرده و در واقع به حداقل میزان خود می رسد و با افزایش  $k$  مقدار خطا نیز افزایش پیدا می کند. برای انتخاب مقدار بهینه  $k$  می توان داده های آموزش و ارزیابی را از مجموعه داده اولیه جدا کرد. سپس با رسم نمودار مربوط به خطای ارزیابی مقدار بهینه  $k$  را انتخاب کرد. این مقدار از  $k$  باید برای همه پیش بینی ها مورد استفاده قرار بگیرد و در واقع برای همه مراحل پیش بینی باید از یک  $k$  مشخص استفاده شود.

## SVM (ماشین بردار پشتیبان)

این روش از جمله روش های نسبتاً جدیدی است که در سال های اخیر کارایی خوبی نسبت به روش های قدیمی تر برای طبقه بندی نشان داده است. SVM یک الگوریتم نظارت شده یادگیری ماشین است که هم برای مسائل طبقه بندی و هم مسائل رگرسیون قابل استفاده است؛ با این حال از آن بیشتر در مسائل طبقه بندی استفاده می شود در الگوریتم SVM، هر نمونه داده را به عنوان یک نقطه در فضای  $n$ -بعدی روی نمودار پراکندگی داده ها ترسیم کرده ( $n$  تعداد ویژگی هایی است که یک نمونه داده دارد) و مقدار هر ویژگی مربوط به داده ها، یکی از مؤلفه های مختصات نقطه روی نمودار را مشخص می کند. سپس، با ترسیم یک خط راست، داده های مختلف و متمایز از یکدیگر را دسته بندی می کند.



بطور کلی، روش کار این الگوریتم به این صورت است که تابع کرنلی را برای استفاده انتخاب می کنیم و سپس مقدار  $C$  و گاما را مشخص می کنیم و از الگوریتم استفاده می نماییم.  $SVM$  یک الگوریتم باینری است. یعنی برای تشخیص ۲ کلاس است و اگر تعداد کلاسها بیشتر باشد، باید همانطور که گفته شد، تابع کرنل مناسب را انتخاب کنیم. این الگوریتم دارای ۴ تابع کرنل معروف است که عبارتند از:

- تابع خطی

$$k(x_i, x_j) = x_i * x_j$$

- تابع چندجمله ای

$$k(x_i, x_j) = (1 + x_i * x_j)^d$$

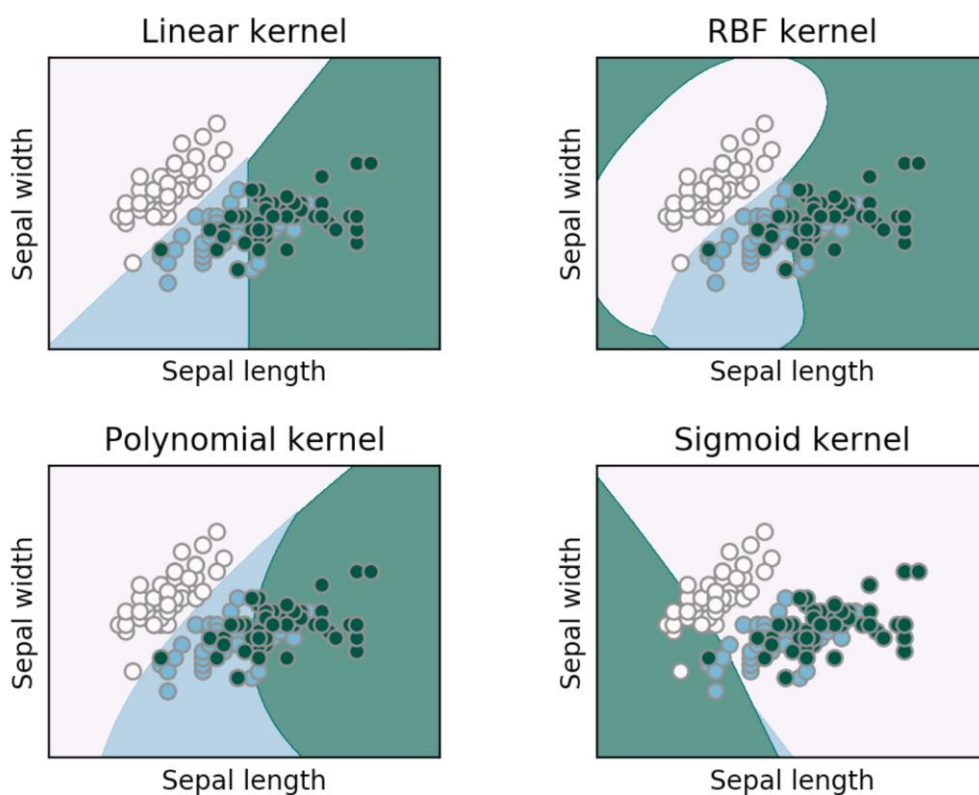
- تابع پایه شعاعی (rbf)

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

- تابع سیگموئید

$$k(x_i, x_j) = \tanh(\alpha x^T y + c)$$

در تصاویر زیر نیز، عملکرد این توابع در برابر تشخیص کلاسها در زمانی که بیش از ۲ کلاس داریم، کنارهم به نمایش آمده اند.

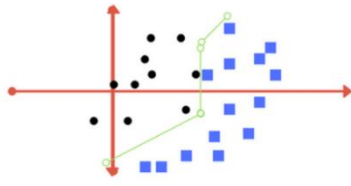


در بین این توابع، تابع پایه شعاعی بهترین عملکرد را خواهد داشت.

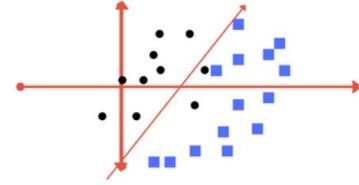
دو پارامتر بسیار مهمی که باید تعیین شوند نیز مقادیر  $C$  و  $\gamma$  می باشند.

پارامتر  $C$ ، نشان می دهد که چقدر می خواهید از داده های کلاسه بندی نشده در هر نمونه ترین اجتناب کنید.

برای مقادیر بزرگ  $C$ ، بهینه سازی یک ابرصفحه با حاشیه کوچک تر را انتخاب می کند در صورتی که این ابر صفحه کار بهتری برای دریافت همه نقاط ترین به درستی طبقه بندی کند. در مقابل، مقدار بسیار کم  $C$  باعث می شود تا این بهینه ساز به دنبال یک ابر صفحه جداساز بزرگ تر باشد، حتی اگر این ابرصفحه نقاط بیشتری را به صورت کلاسه بندی نشده نشان دهد.

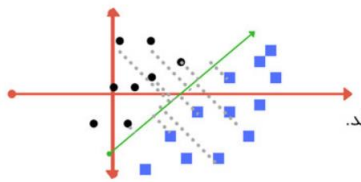


مقدار بالا C



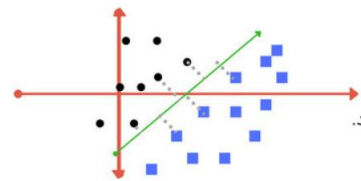
مقدار پایین C

پارامتر گاما تعیین می کند که تاثیر هر مثال آموزشی تا چه حد باشد، که مقادیر کم آن به معنی تاثیر بیشتر و مقادیر زیاد آن به معنی تاثیر کمتر است. به عبارت دیگر، با گاما کم، نقاط دور از خط احتمالی جدا سازی برای محاسبه خط جدا ساز در نظر گرفته می شوند. و با گاما بالاتر، نقاط نزدیک به خط احتمالی در محاسبه در نظر گرفته می شوند.



**گاما پایین**

تنها نقاط دور  
در نظر گرفته می شوند.



**گاما بالا**

تنها نقاط نزدیک  
در نظر گرفته می شوند.

با بررسی ها و خواندن مقالات و کار با این الگوریتمها به همراه مجموعه داده های بدست آمده، به این نتیجه رسیدیم که SVM مناسب ترین الگوریتم برای این پروژه می باشد.

حال به بررسی پیاده سازی این الگوریتم در این پروژه می پردازیم.

بطور کلی قسمت یادگیری ماشین پروژه با زبان پایتون پیاده سازی شده است. دلیل انتخاب این زبان وجود کتابخانه های قدرتمند در زمینه کار و پردازش داده و یادگیری ماشین بوده است.

ساختار این بخش از پروژه بصورت زیر می باشد:

- فایل پایتون برای پیش پردازش بر روی داده های خام با نام preprocess.py
- فایل پایتون برای تبدیل دیتاست های موجود به یک دیتاست جامع با نام concatenate.py
- فایل پایتون برای آموزش دادن به الگوریتم SVM با نام svm.py
- فایل پایتون برای پیش بینی کردن حرکت فرد با نام main.py

### کتابخانه های استفاده شده در قسمت یادگیری ماشین پروژه

**Pandas:** یک کتابخانه متن باز با گواهینامه BSD است که کارایی بالا، ساختاری با قابلیت استفاده آسان و ابزارهای تحلیل داده برای زبان برنامه نویسی پایتون را فراهم می کند. در واقع، می توان گفت پانداس یک کتابخانه قدرتمند برای تحلیل، پیش پردازش و بصری سازی داده ها است.

**Skicit Learn:** یکی از چندین کتابخانه ایست که برای کاربردهای یادگیری ماشین ایجاد شده است. این کتابخانه شامل پیاده سازی الگوریتم های یادگیری نظارت شده و نظارت نشده است. Skicit learn کتابخانه ایست که بر روی ماژول دیگری به نام SciPy نوشته شده است.

### توضیح کدهای پایتون مربوط به الگوریتم SVM

#### • فایل svm.py:

ابتدا فایل دیتاست را خوانده و سپس داده های گم شده را حذف می کنیم.

```
# Read csv
df = pd.read_csv('dataset.csv')

# Remove missing dataset
for index, row in df.iterrows():
    for x in row:
        if type(x) == str:
            continue
        if math.isnan(x):
            df = df.drop(index)
            break
```

سپس عنوان هر داده و بقیه اطلاعاتش را از هم جدا نموده و ۷۰٪ داده ها را برای آموزش و ۳۰٪ را برای تست انتخاب می کنیم.

```
# separate label and other columns into two df
X = df.drop('type', axis=1).copy()
y = df['type'].copy()

# centering and scaling
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
X_train_scaled = scale(X_train)
X_test_scaled = scale(X_test)
```

بعد از آن الگوریتم svm را تنظیم نموده و داده ها را به آن می دهیم.

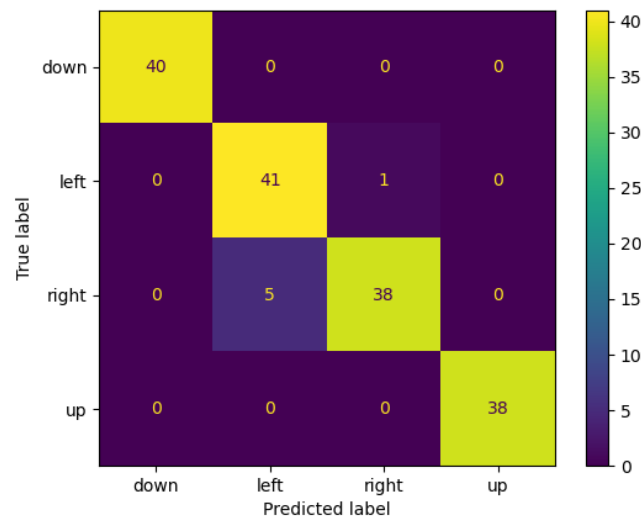
```
# build a preliminary SVM
params_grid = [
    {
        'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
        'C': [1, 10, 100, 1000]
    },
    {
        'kernel': ['linear'],
        'C': [1, 10, 100, 1000]
    }
]
clf_svm = GridSearchCV(SVC(random_state=42), params_grid)
clf_svm.fit(X_train_scaled, y_train)
prediction = clf_svm.predict(X_test_scaled)
print(accuracy_score(y_test, prediction))
```

سپس نمره دقت را پرینت می کنیم که ۹۶.۵٪ است.

```
0.9657142857142857
```

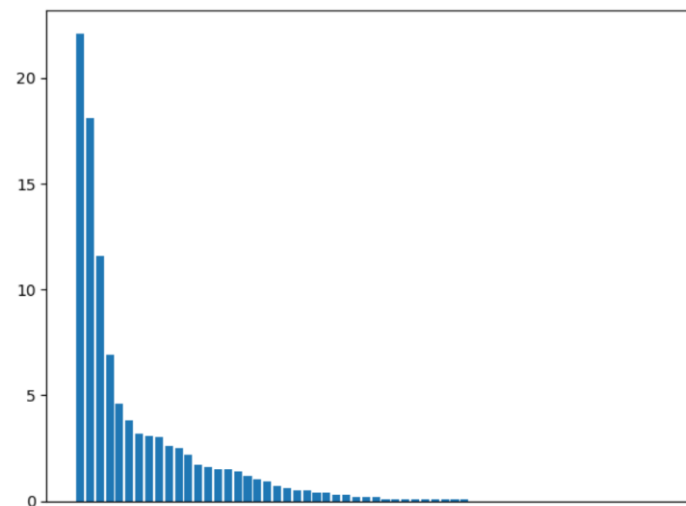
ماتریس درهم ریختگی را هم رسم می کنیم.

```
# plot confusion matrix
plot_confusion_matrix(clf_svm,
                      X_test_scaled,
                      y_test)
print(plot_confusion_matrix)
plt.show()
```



در بخش بعد، میزان تاثیر هر ویژگی به همان ترتیب که در ستون ها هستند را نشان می دهد.

```
pca = PCA()
X_train_pca = pca.fit_transform(X_train_scaled)
per_var = np.round(pca.explained_variance_ratio_*100, decimals=1)
labels = [str(x) for x in range(1, len(per_var) + 1)]
plt.bar(x=range(1, len(per_var) + 1), height=per_var)
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False
)
plt.show()
```



در آخر نیز میزان دقت را بصورت دستی نیز محاسبه نموده که به همان عدد قبلی میرسیم.

```
counter = 0
right_counter = 0
down_counter = 0
left_counter = 0
up_counter = 0

predicted = clf_svm.predict(X_test_scaled)
temp = zip(predicted, y_test)

correct_answer = 0
wrong_answer = 0
for x, y in temp:
    if x == 'right':
        if y == 'right':
            correct_answer += 1
        else:
            wrong_answer += 1
    right_counter += 1
```



```

elif x == 'up':
    if y == 'up':
        correct_answer += 1
    else:
        wrong_answer += 1
    up_counter += 1
elif x == 'left':
    if y == 'left':
        correct_answer += 1
    else:
        wrong_answer += 1
    left_counter += 1
elif x == 'down':
    if y == 'down':
        correct_answer += 1
    else:
        wrong_answer += 1
    down_counter += 1
counter += 1
print(right_counter)
print(up_counter)
print(down_counter)
print(left_counter)
print(correct_answer)
print(wrong_answer)
print(correct_answer/len(X_test))

```

در نهایت نیز مدل بدست آمده را برای استفاده های بعدی در فایل ذخیره می کنیم.

```

# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(clf_svm, open(filename, 'wb'))

```

این فایل، در `main.py` استفاده شده است.

در فایل `main.py`، فایل `finalized_model` را `load` می کنیم و ورودی حرکت را به این فایل می دهیم. ( این ورودی در برنامه دسکتاپ تشخیص حرکت به این فایل داده می شود که در بخش مربوطه توضیح داده می شود.) و سپس داده های گرفته شده ی حسگرها را نرمال می کنیم.

```

filename = 'C:/Users/Amirhossein/PycharmProjects/bachelor-
project/finalized_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))

path = input("#f-->")
df = pd.read_csv(path)
gyroscope_df = df.loc[df['ACC_or_GYR'] == 'GYR'].reset_index(drop=True)
accumulator_df = df.loc[df['ACC_or_GYR'] == 'ACC'].reset_index(drop=True)

# normalized gyro
gyro_max = gyroscope_df[['x', 'y', 'z']].max()

```

```
df_max = gyro_max.max()
gyroscope_df = gyroscope_df[['x', 'y', 'z']] / df_max

# normalized acc
acc_max = accumulator_df[['x', 'y', 'z']].max()
df_max = acc_max.max()
accumulator_df = accumulator_df[['x', 'y', 'z']] / df_max
```

در ادامه فایل نیز ویژگی را استخراج نموده و حرکت را تشخیص می دهد و در برنامه دسکتاپ نمایش داده می شود که در ادامه توضیح داده شده است.

## فصل چهارم

### پیاده سازی برنامه دسکتاپ

در این پروژه، ۲ برنامه دسکتاپ پیاده سازی شده اند که توسط Qt پیاده سازی شده است. Qt در واقع مجموعه ای از کتابخانه ها و Header های نوشته شده با زبان C++ هست. این Library ها امکان استفاده آسان از شبکه، گرافیک، پایگاه داده و... را به برنامه نویس می دهند. همچنین این فریمورک قابلیت توسعه نرم افزار هایی با رابط کاربری گرافیکی (GUI) نیز دارد و میتوان علاوه بر برنامه های Console، برنامه های گرافیکی را نیز به آسانی توسعه داد. با استفاده از Qt میتوان برای ویندوز، لینوکس، اندروید، iOS، OS X، سیستم های تعبیه شده (Embedded system) و... نرم افزارهای مختلفی توسعه داد. Qt از Signal ها و Slot ها برای رسیدگی آسان به رویدادها استفاده می کند. Signal و Slot یک واحد زبانی مورد استفاده در Qt که بین اشیا مختلف ارتباط برقرار می کند. این کار اجرای observer pattern را راحت می کند و در عین حال از ایجاد کد boilerplate جلوگیری می کند. مفهوم کلی به این صورت است که ویجت های رابط کاربری گرافیکی می توانند سیگنال هایی را ارسال کنند که حاوی اطلاعات مربوط به رویداد هاست که این اطلاعات توسط کنترل های دیگر و به کمک توابعی ویژه که به آن ها شکاف ها گفته می شود دریافت می شوند.

### ماژول های ضروری Qt

- Qt Core: تنها ماژول ضروری Qt که حاوی دسته هایی است که توسط سایر ماژول ها مورد استفاده قرار می گیرند. این دسته ها شامل سیستم هم زمانی و threading، محفظه ها، سیستم رویداد ها، پلاگین ها و امکانات مربوط به ورودی و خروجی می شود.
- Qt GUI: ماژول مرکزی رابط کاربری
- Qt Widgets: حاوی دسته هایی برای نرم افزار های کاربردی کلاسیک دارای رابط های کاربری گرافیکی بر پایه ویجت ها و دسته های QGraphicsScene.
- Qt QML: ماژولی برای زبان های QML و JavaScript
- Qt Quick: ماژول نرم افزار های کاربردی دارای رابط کاربری گرافیکی که با استفاده از QML2 نوشته شده اند.
- Qt Quick Controls: کنترل های شبیه به ویجت برای Qt Quick که برای نرم افزار های دسکتاپ در نظر گرفته شده است.
- Qt Quick Layouts: طرحی برای مرتب کردن اجزا در Qt Quick
- Qt Network: لایه انتزاعی شبکه به همراه پشتیبانی از TCP, UDP, HTTP, SSL و از نسخه ۵.۳ به بعد با پشتیبانی از SPDY

- Qt Multimedia: دسته هایی برای کارایی های صوتی، تصویری، رادیویی و دوربین
- Qt Multimedia Widgets: ویجت های Qt Multimedia
- Qt SQL: حاوی دسته هایی برای یکپارچه سازی بانک اطلاعاتی با استفاده از SQL
- Qt WebEngine: مجموعه جدیدی از API های Qt Widget و QML webview بر پایه Chromium
- Qt Test: دسته هایی برای تست واحد نرم افزار ها و کتابخانه های Qt

با Qt چه نرم افزار های معروفی توسعه داده شده اند؟

Bitcoin Core (یک Wallet برای بیت کوین)

Google Earth

Adobe Photoshop Albums و Adobe Photoshop Elements

Skype

Spotify (نسخه لینوکس)

Telegram Desktop (تلگرام برای ویندوز ، لینوکس و مک)

VLC media player

WPS Office

Wireshark (آنالیز کننده packet های شبکه)

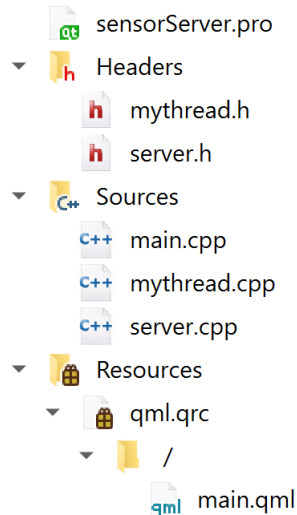
VirtualBox (نرم افزاری برای شبیه سازی سیستم عامل ها)

AMD Radeon Software Crimson

### برنامه اول: دریافت اطلاعات حسگرهای شتاب سنج وژیروسکوپ از برنامه موبایل

بطور کلی، در این برنامه اطلاعات این ۲ حسگر را گرفته و در فایل CSV ذخیره می شود.

چهارچوب کلی فایل های این برنامه بصورت زیر می باشد.



اولین فایلی که بررسی می شود، **sensorServer.pro** است که تنظیمات پروژه در آن قرار دارد.

در خط اول ماژول های **quick** و **network** اضافه شده اند.

```
QT += quick network
```

در خط بعدی کامپایلر مورد استفاده **C++** مشخص شده است.

```
CONFIG += c++11
```

در خط بعد ماکرو مربوط به نشان دادن اخطار در مشکلات **API** می باشد.

```
DEFINES += QT_DEPRECATED_WARNINGS
```

در بخش بعدی فایل های **C++**، **qml** و آیکون برنامه مشخص شده اند.

```
SOURCES += \
    main.cpp \
    mythread.cpp \
    server.cpp
```

```
RESOURCES += qml.qrc
RC_ICONS = icon.ico
```

در نهایت نیز نحوه اجرا شدن پروژه و هدر ها مشخص شده اند.

```
# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
    !isEmpty(target.path): INSTALLS += target

HEADERS += \
mythread.h \
server.h
```

فایل بعدی، main.cpp است که فایلی است که اول از همه اجرا می شود. در این فایل، سرور ایجاد می شود و Qt را به c++ متصل می کند.

سپس qml را متصل نموده.

و پس از آن، qml را اجرا می نمایم.

```
int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);

    QGuiApplication app(argc, argv);

    //-- painter class --//
    qmlRegisterType<Server>("com.Server", 1, 0, "Server");

    QQmlApplicationEngine engine;
    const QUrl url(QStringLiteral("qrc:/main.qml"));
    QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,
    &app, [url](QObject *obj, const QUrl &objUrl) {
        if (!obj && url == objUrl)
            QCoreApplication::exit(-1);
    }, Qt::QueuedConnection);
    engine.load(url);

    return app.exec();
}
```

فایل بعدی، `server.h` میباید که هدر بخش سرور برنامه است.

در بخش اول، یک سرور سوکت از نوع TCP تعریف می کنیم که تلفن همراه به آن متصل شود. درون این کلاس پراپرتی `serverStatus` تعریف می کنیم که وضعیت سرور را بیان می کند و تمامی دسترسی ها می دهیم تا در فایل `qml` که در آینده توضیح داده می شود.

```
class Server : public QTcpServer
{
    Q_OBJECT
    Q_PROPERTY(bool serverStatus READ isServerStatus WRITE setServerStatus
                NOTIFY serverStatusChanged)
```

در قسمت بعدی، تابع های پابلیک تعریف شده است.

`Explicit Server`، تابع سازنده کلاس `server` است.

دو تابع دیگر نیز تعریف شده اند که کاربردشان در فایل `C++` تعریف شده است.

در بخش بعد نیز ۲ سیگنال `serverStatusChanged` و `gyroscopeData` تعریف شده اند.

در بخش بعد نیز `slot` ها تعریف شده اند.

در بخش توابع پرایوت، ابتدا یک اشاره گر از سرور سوکت ساخته شده و یک تابع `serverStatus`

و در آخر نیز یک تابع محافظت شده `incomingConnection` که تابع خود `Qt` است را تعریف می کنیم که می خواهیم `override` شود.

```
public:
    explicit Server(QObject *parent = nullptr);

    bool isServerStatus();
    void setServerStatus(bool serverStatus);

signals:
    void serverStatusChanged();

public slots:
    void startServer();
    void closeServer();
    QString getIpAddress();

private:
    QTcpServer *server;
    bool serverStatus;

protected:
    void incomingConnection(qintptr socketDescriptor) override;

};
```

فایل بعدی، server.cpp می باشد.

در ابتدا تابع سازنده سرورمان ساخته می شود.

```
Server::Server(QObject *parent) : QTcpServer(parent)
{
    serverStatus = false;
}
```

سپس تابع startServer تعریف شده است که که بر روی آی پی محلی و پورت ۱۲۳۴ سرور را اجرا می کنیم. اگر نتواند اجرا کند، serverStatus=false می ماند و اگر متصل شود، serverStatus=true می شود. سپس با دستور emit خبر می دهیم که serverStatus تغییر کرده است.

```
void Server::startServer()
{
    if(!this->listen(QHostAddress::Any, 1234)){
        qDebug() << "could not start server";
        serverStatus = false;
        emit serverStatusChanged();
    }
    else{
        qDebug() << "Listening...";
        serverStatus = true;
        emit serverStatusChanged();
    }
}
```

تابع بعدی closeServer می باشد که سرور باز شده را می بندد. سپس با دستور emit خبر می دهیم که serverStatus تغییر کرده است.

```
void Server::closeServer()
{
    if(this->serverStatus==true){
        qDebug() << "closing server";
        serverStatus = false;
        emit serverStatusChanged();
        this->close();
    }
}
```

تابع بعدی isServerStatus می باشد که باز یا بسته بودن سرور را نشان می دهد.

```
bool Server::isServerStatus()
{
    return this->serverStatus;
}
```



در تابع بعدی آی پی محلی سیستم را گرفته تا به کاربر نمایش دهیم و در اپلیکیشن موبایل وارد کند.

```
QString Server::getIpAddress()
{
    qDebug() << "get ip...";
    QString ip_address = "";
    const QHostAddress &localhost = QHostAddress(QHostAddress::LocalHost);
    for (const QHostAddress &address: QNetworkInterface::allAddresses()) {
        if (address.protocol() == QAbstractSocket::IPv4Protocol && address
            != localhost)
        {
            ip_address = address.toString();
            qDebug() << "ip: " << ip_address;
        }
    }
    return ip_address;
}
```

در تابع بعدی، وضعیت سرور را ست می کنیم که روشن است یا خاموش.

```
void Server::setServerStatus(bool serverStatus)
{
    this->serverStatus = serverStatus;
}
```

حال تابع incomingConnection را override می کنیم. در این قسمت هر دستگاهی که متصل می شود با socketDescriptor مشخص می شود و برای هر کدام از دستگاه های متصل شده، یک thread جداگانه ایجاد می شود و پس از اتمام کار با آن دستگاه و قطع شدن آن، thread را از بین می بریم.

```
void Server::incomingConnection(qintptr socketDescriptor)
{
    qDebug() << socketDescriptor << " Connecting...";
    MyThread *thread = new MyThread(socketDescriptor, this);
    connect(thread, SIGNAL(finished()), thread, SLOT(deleteLater()));
    thread->start();
}
```

مباحث thread در ۲ فایل mythread.h و mythread.cpp پیاده سازی شده اند.

در فایل هدر، کلاسی میسازیم که از Qtthread خود Qt ارث بری کرده است.

```
class MyThread : public QThread
{
    Q_OBJECT
```

در بخش `public`، تابع سازنده `thread` و تابع `run` که از توابع خود `Qthread` است، تعریف شده اند. دلیل تعریف در بخش `public` این است که از بیرون می‌خواهد `thread` را اجرا کند.

```
public:
    explicit MyThread(int ID, QObject *parent = nullptr);
    void run();
```

۲ سیگنال `recvData` و `error` تعریف شده اند.

```
signals:
    void error(QTcpSocket::SocketError socketerror);
    void recvData(int socketDescriptor, QString data);
```

۲ اسلات `readyRead` و `disconnected` تعریف شده اند.

```
public slots:
    void readyRead();
    void disconnected();
```

یک اشاره گر به کلاس `thread`، یک عدد که قبل تر گفته شد که نشان دهنده ی دستگاه هابیت که متصل هستند با نام `socketDescriptor` و یک لیست با نام `buffer` را بصورت `private` تعریف شده اند.

```
private:
    QTcpSocket *socket;
    int socketDescriptor;
    QStringList buffer;

};
```

در فایل `mythread.cpp`، ابتدا سازنده کلاس `MyThread` را تعریف می کنیم که در آن آیدی مربوط به موبایل مربوطه را مشخص می کنیم.

```
MyThread::MyThread(int ID, QObject *parent)
    : QThread(parent)
{
    this->socketDescriptor = ID;
}
```

سپس در تابع `run`، یک شیء از سوکت می سازیم و تلفن همراه را به آن متصل می کنیم و اگر در این فرآیند مشکلی پیش بیاید و نتواند متصل شود، به سیگنال `error` اشاره می شود.

در خط بعدی برای اینکه چک کنیم چه زمانی اطلاعات از سوکت می آید و هر لحظه چک نکنیم، از سیگنال استفاده می کنیم تا هر زمانی که اطلاعات بر روی سوکت آمد، اطلاع پیدا کند و تابع `readyRead` را فراخوانی کند.

بعد از آن نیز بررسی می کنیم هر زمان که از سمت تلفن همراه اتصال قطع شد، **thread** هم بسته شود.

```
void MyThread::run()
{
    // thread start here

    qDebug() << socketDescriptor << " Starting Thread ..." ;
    socket = new QTcpSocket();
    if(!socket->setSocketDescriptor(this->socketDescriptor)){

        emit error(socket->error());
        return;
    }

    connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()),
Qt::DirectConnection);
    connect(socket, SIGNAL(disconnected()), this, SLOT(disconnected()),
Qt::DirectConnection);

    qDebug() << socketDescriptor << " client connected" ;

    exec();
}
```

در تابع **readyRead**، تمامی اطلاعات را از سوکت می خوانیم و فرمتش را تبدیل به یونیکد می کنیم و هر داده را از داده ی دیگر با نماد هشتگ جدا می کنیم و اطلاعات را درون لیست می ریزیم.

```
void MyThread::readyRead()
{
    QByteArray Data = socket->readAll();

    QString DataAsString = QTextCodec::codecForMib(106 )->toUnicode(Data);

    QStringList list = DataAsString.split("#");

    for(int i=0; i< list.length(); i++){
        if(list.at(i).length() > 0){

            buffer.append(list.at(i));
        }
    }
}
```

تابع **disconnected** نیز به این صورت است که هنگامی که اتصال قطع می شود، فایل **CSV** با نام تاریخ و ساعت همان لحظه می سازیم و سپس برای نوشتن بر روی این فایل، یک استریم ساخته و در خط نام ستون های فایل **CSV** را ذخیره کرده و اطلاعات را از لیست خوانده و به عنوان سطرهای فایلمان ذخیره می کنیم و در نهایت سوکت را از بین می برد.

```
void MyThread::disconnected()
{
    qDebug() << socketDescriptor << " disconnected" ;

    QDateTime d;
    QString name = d.currentDateTime().toString("yyyy_mm_dd-hh_mm_ss_zzz");
    qDebug() << "name : " << name;
    QFile file(name + ".csv");
```

```

if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {
    qDebug() << "can not open file";
    return;
}
QTextStream out1(&file);
out1.setDevice(&file);

    out1 << "ACC_or_GYR,millisecond,x,y,z" << "\n";

for(int i=0; i< buffer.length(); i++){

    QStringList t = buffer.at(i).split("_");
    out1 << t.join(',') << "\n";
}

qDebug() << "file closed, size" << file.size() ;
file.close();
socket->deleteLater();
buffer.clear();

    exit(0);
}

```

فایل بعدی که بررسی می شود، فایل qml است که بخش رابط کاربری و گرافیکی در آن پیاده سازی شده است.

ابتدا سایز پنجره اپلیکیشن و رنگ پس زمینه و نام نمایشی برنامه را مشخص می کنیم. و بعد از آن، سروری که ساخته بودیم را مشخص می کنیم و `serverStatus` را `false` می کنیم که یعنی در اول اجرا برنامه، سرور بسته است.

```

Window {
    visible: true
    width: 500
    height: 150
    color: "#f5f5f5"
    title: qsTr("Sensor Server")

    Server{
        id: server
        serverStatus: false
    }
}

```

سپس یک سطر مشخص می کنیم و دکمه ای با آیدی `btn_startSewrver` می سازیم که زمانی که سرور قطع باشد، متن دکمه `start server` می باشد و زمانی که سرور وصل باشد، `close server` می باشد.

```
ColumnLayout{
    anchors.fill: parent

    //-- tools --//
    Item{
        Layout.fillWidth: true
        Layout.preferredHeight: btn_startSewrver.implicitHeight * 1.5

        RowLayout{
            anchors.fill: parent
            anchors.leftMargin: 170

            Button{
                id: btn_startSewrver
                text: "start server"
                background: Rectangle {
                    implicitWidth: 100
                    implicitHeight: 40
                    color: "#FFA500"
                    border.color: "#000000"
                    border.width: 1
                    radius: 4
                }
                onClicked: {
                    if(!server.serverStatus){
                        server.startServer()
                        btn_startSewrver.text = 'close server'
                    }
                    else {
                        server.closeServer()
                        btn_startSewrver.text = 'start server'
                    }
                }
            }
        }
    }
}
```

در کنار این دکمه، دایره ای را قرار می دهیم که وضعیت قطع یا وصل بودن سرور را نشان می دهد. اگر سرور وصل باشد، سبز است و در غیر این صورت قرمز می شود.

```
Rectangle{
    Layout.preferredHeight: btn_startSewrver.height * 0.5
    Layout.preferredWidth: height
    radius: width * 0.5
    border.color: "black"
    border.width: 1
    smooth: true
    color: server.serverStatus ? "Green" : "Red"
}
```

و در کنار این دایره نیز، آی پی محلی سیستم را نمایش می دهیم.

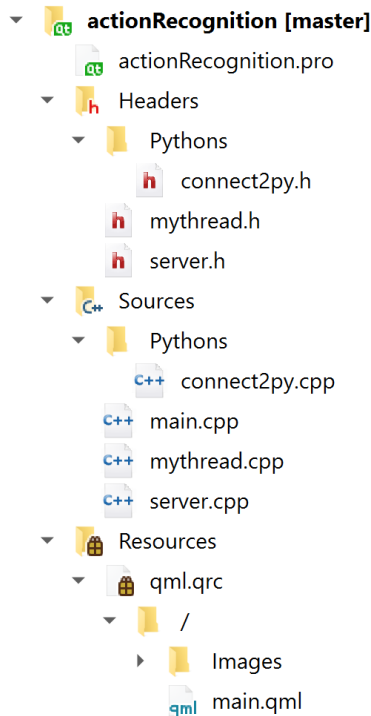
```
Label{  
    id: lbl_serverStatus  
    text: server.getIpAddress()  
}
```

خروجی این برنامه نیز به صورت زیر می باشد.



## برنامه دوم: برنامه تشخیص حرکت ورودی

چهارچوب کلی این برنامه مانند برنامه قبل است. تنها بخش جدید، قسمت اتصال به پایتون می باشد که توسط آقای اسدی پیاده سازی شده است.



تنها بخش متفاوت `actionRecognition.pro` با `sensorServer.pro` ، هدر ها و سورس ها هستند که بخش های مربوط به پایتون نیز اضافه شده اند.

```
SOURCES += \
    Pythons/connect2py.cpp \
    main.cpp \
    mythread.cpp \
    server.cpp
```

```
HEADERS += \
    Pythons/connect2py.h \
    mythread.h \
    server.h
```

در فایل `main.cpp` نیز تنها تغییر اضافه شدن بخش `C++` اتصال به پایتون، برای اتصال به `qml` می باشد.

```
qmlRegisterType<Server>("com.Server", 1, 0, "Server");
qmlRegisterType<Connect2Py>("com.Python", 1, 0, "Python");
```

در فایل `server.h` یک سیگنال با نام `captureAction` و یک اسلات با نام `handleCaptureAction` ایجاد شده اند که کاربردشان در ادامه توضیح داده خواهد شد و بقیه فایل مانند برنامه قبل می باشد.

```
signals:
    void captureAction(QString fileUrl);
public slots:
    void handleCaptureAction(QString fileUrl);
```

در فایل `server.cpp`، `handleCaptureAction` پیاده سازی شده است که از سمت فایل های مربوط به `thread`، هنگامی که ارتباط سوکت قطع می شود، سیگنال فعال می شود و یک سیگنال به سمت `qml` میفرستد و در `qml`، فایل پایتونی که قرار است اجرا کند را فراخوانی می کند.

```
void Server::handleCaptureAction(QString fileUrl)
{
    emit captureAction(fileUrl);
}
```

در فایل `mythread.h`، سیگنال `captureAction` تعریف شده است که استفاده اش در خطوط بالا بیان شد.

```
signals:
    void captureAction(QString fileUrl);
```

در تابع `disconnected` فایل `mythread.cpp` نیز در انتها این تابع، سیگنال `captureAction` را فعال کردیم تا کاری را که در فایل `Server.cpp` توضیح داده شد را انجام دهد.

```
void MyThread::disconnected()
{
    qDebug() << socketDescriptor << " disconnected" ;

    QDateTime d;
    QString name = QString::number(d.currentMsecsSinceEpoch());
    qDebug() << "name : " << name;
    QFile file(name + ".csv");
    if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        qDebug() << "can not open file";
        return;
    }

    QTextStream out1(&file);
    out1.setDevice(&file);

    out1 << "ACC_or_GYR,millisecond,x,y,z" << "\n";

    for(int i=0; i< buffer.length(); i++){

        QStringList t = buffer.at(i).split("_");
        out1 << t.join(',') << "\n";
    }
}
```



```

    }

    qDebug() << "file closed, size" << file.size() ;
    file.close();
    socket->deleteLater();
    buffer.clear();

    qDebug() << "current dir: " << QDir::currentPath();
    QString url = QDir::currentPath() + "/" + name + ".csv";
    emit captureAction(url);

    exit(0);
}

```

در فایل qml ابتدا پنجره اصلی برنامه را می سازیم و برای هر حرکت بالا، پایین، چپ و راست، یک عدد را به عنوان پراپرتی تعریف می کنیم.

```

Window {
    id: root

    property int _ACTION_UP      : 1
    property int _ACTION_LEFT   : 2
    property int _ACTION_RIGHT  : 3
    property int _ACTION_DOWN   : 4

    property int current_action: 0
    onCurrent_actionChanged: {
        print("current action is " , current_action)
    }

    visible: true
    width: 750
    height: 480
    title: qsTr("Action Detection")
}

```

سپس سروری را که به qml متصل شده بود را مشخص می کنیم.

```

Server{
    id: server
    serverStatus: false
}

```

در بخش بعدی قسمت اتصال به پایتون را می نویسیم که پس از تشخیص حرکت، انیمیشن مربوط به حرکت را اجرا می کند.

```

Python{
    id: py

    //-- QString action --//
    onDetectAction: {
        if(action.indexOf("UP") !== -1){

```

```

        current_action = _ACTION_UP
        anim_up.restart()
    }
    else if(action.indexOf("DOWN") !== -1){
        current_action = _ACTION_DOWN
        anim_down.restart()
    }
    else if(action.indexOf("RIGHT") !== -1){
        current_action = _ACTION_RIGHT
        anim_right.restart()
    }
    else if(action.indexOf("LEFT") !== -1){
        current_action = _ACTION_LEFT
        anim_left.restart()
    }
}
}
}

```

آی پی سیستم را که در فایل C++ تشخیص داده شده بود را نیز به کاربر نمایش می دهیم.

```

Component.onCompleted: {
    var ips = server.getServerIp()
    var list_ip = ips.split('_')
    print("ipos: " + list_ip)
    for(var i=0; i<list_ip.length; i++){

        var secs = list_ip[i].split('.')

        if(secs[3] === "1") continue

        lbl_ip.text = lbl_ip.text + list_ip[i]
        if(i !== list_ip.length-1) lbl_ip.text = lbl_ip.text + " , "
    }
}

```

و در کنار آی پی ها یک تصویر قرار گرفته شده است.

```

RowLayout{

    anchors.fill: parent
    anchors.margins: 3

    Image {
        source: "qrc:/Images/ips.png"

        fillMode: Image.PreserveAspectFit
        sourceSize.width: parent.height * 0.5
        sourceSize.height: parent.height * 0.5
    }

    Label{
        id: lbl_ip
        text: ""
    }
    Item{ Layout.fillWidth: true }
}

```

سپس دکمه ای را برای قطع و وصل کردن سرور می گزاریم و تصویری در کنارش قرار می دهیم که وضعیت سرور را مشخص کند و در کنارش نیز متن قرار دارد.

```
//-- start/stop server --//
Button{
    id: btn_startSewrver
    text: "Start"
    onClicked: {

        if(!server.serverStatus){

            server.startServer()
            btn_startSewrver.text = "Stop"
        }
        else{
            server.stopServer()
            btn_startSewrver.text = "Start"
        }
    }
}

//-- image for server status --//
Image {
    id: img_server
    source: server.serverStatus ?
"qrc:/Images/server_on.png" : "qrc:/Images/server_off.png"
    //          Layout.fillHeight: true
    fillMode: Image.PreserveAspectFit
    sourceSize.width: parent.height * 0.8
    sourceSize.height: parent.height * 0.8
}

//-- server status text --//
Label{
    id: lbl_serverStatus
    text: server.serverStatus ? "Listening..." : "Server
closed."
}
```

در کنار این موارد نیز یک دکمه قرار گرفته است که با فشردن آن یک ورودی باز شده و آدرس فایل اجرایی پایتون را می گیرد.

```
//-- set python script --//
Item{
    Layout.fillWidth: true
    Layout.fillHeight: true

    RowLayout{
        anchors.fill: parent
        anchors.margins: 3
    }
}
```

```

Item{Layout.fillWidth: true}

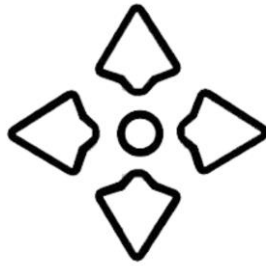
Button{
    Layout.preferredWidth: implicitHeight
    text: ""
    icon.source: "qrc:/Images/folder.svg"

    Tooltip.delay: 600
    Tooltip.timeout: 5000
    Tooltip.visible: hovered
    Tooltip.text: qsTr("set python main.py run
script text")

    onClicked: {
        popup.open()
    }
}
}
}

```

در بخش بعدی، جهت ها پیاده سازی شده اند که بصورت زیر هستند.



برای پیاده سازی این تصویر، ابتدا مستطیلی که این تصویر درون آن قرار می گیرد را تعریف می کنیم و سپس هر کدام از تصویر های مرکز، بالا، راست، چپ و پایین پیاده سازی شده اند. به این ترتیب که زمانی که حرکتی تشخیص داده نشده است، تمامی جهت ها به رنگ مشکی هستند و هنگامی که حرکتی تشخیص داده می شود، جهت حرکت به رنگ بنفش درآمده و همراه با انیمیشن حرکت می کند.

```

/-- action images -/--
Item{
    Layout.fillWidth: true
    Layout.fillHeight: true

    Rectangle{
        anchors.fill: parent
        anchors.margins: 10
        color: "transparent"
        border{
            width: 1
            color: "grey"
        }
    }
}

```

```

    //-- center --//
    Image {
        source: "qrc:/Images/action/action_center.png"
        anchors.centerIn: parent
    }

    //-- up --//
    Image {
        id: img_up
        source: current_action === _ACTION_UP ?
"qrc:/Images/action/action_up_on.png" :
"qrc:/Images/action/action_up_off.png"
        anchors.centerIn: parent

        SequentialAnimation{
            id: anim_up
            NumberAnimation {
                target: img_up
                property: "anchors.verticalCenterOffset"
                from: 0; to: -animXdiff;
                duration: 200
                easing.type: Easing.InOutQuad
            }
            NumberAnimation {
                target: img_up
                property: "anchors.verticalCenterOffset"
                from: -animXdiff; to: 0;
                duration: 200
                easing.type: Easing.InOutQuad
            }
        }
    }

    //-- right --//
    Image {
        id: img_right
        source: current_action === _ACTION_RIGHT ?
"qrc:/Images/action/action_right_on.png" :
"qrc:/Images/action/action_right_off.png"
        anchors.centerIn: parent

        SequentialAnimation{
            id: anim_right
            NumberAnimation {
                target: img_right
                property: "anchors.horizontalCenterOffset"
                from: 0; to: animXdiff;
                duration: 200
                easing.type: Easing.InOutQuad
            }
            NumberAnimation {
                target: img_right
                property: "anchors.horizontalCenterOffset"
                from: animXdiff; to: 0;
                duration: 200
                easing.type: Easing.InOutQuad
            }
        }
    }

    //-- left --//

```

```

        Image {
            id: img_left
            source: current_action === _ACTION_LEFT ?
"qrc:/Images/action/action_left_on.png" :
"qrc:/Images/action/action_left_off.png"
            anchors.centerIn: parent

            SequentialAnimation{
                id: anim_left

                NumberAnimation {
                    target: img_left
                    property: "anchors.horizontalCenterOffset"
                    from: 0; to: -animXdiff;
                    duration: 200
                    easing.type: Easing.InOutQuad
                }
                NumberAnimation {
                    target: img_left
                    property: "anchors.horizontalCenterOffset"
                    from: -animXdiff; to: 0;
                    duration: 200
                    easing.type: Easing.InOutQuad
                }
            }
        }

    //-- down --//
    Image {
        id: img_down
        source: current_action === _ACTION_DOWN ?
"qrc:/Images/action/action_down_on.png" :
"qrc:/Images/action/action_down_off.png"
        anchors.centerIn: parent

        SequentialAnimation{
            id: anim_down

            NumberAnimation {
                target: img_down
                property: "anchors.verticalCenterOffset"
                from: 0; to: animXdiff;
                duration: 200
                easing.type: Easing.InOutQuad
            }
            NumberAnimation {
                target: img_down
                property: "anchors.verticalCenterOffset"
                from: animXdiff; to: 0;
                duration: 200
                easing.type: Easing.InOutQuad
            }
        }
    }
}

```

در آخر نیز پاپ آپی که بعد از فشردن دکمه ای که برای وارد کردن آدرس فایل پایتون است را تعریف می کنیم و بصورت پیش فرض آدرسی در آن قرار گرفته شده است که قابل تغییر می باشد.

```
//-- set python main.py run script --//
Popup {
    id: popup

    x: root.width/2 - width/2
    y: root.height/2 - height/2
    width: 600
    height: 100
    modal: true
    focus: true
    closePolicy: Popup.CloseOnEscape | Popup.CloseOnPressOutside

    RowLayout{
        anchors.fill: parent
        anchors.margins: 5

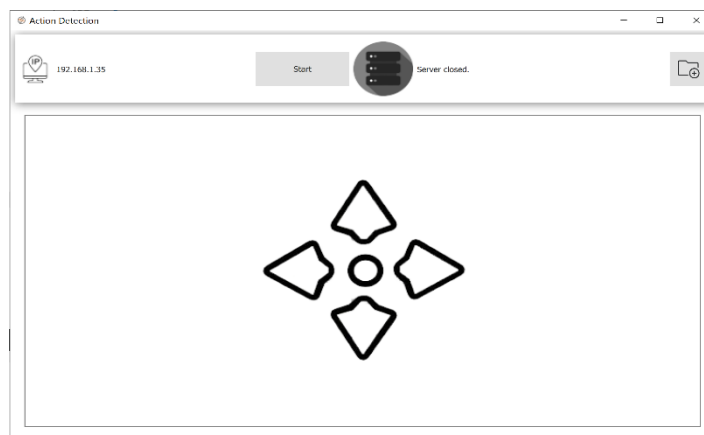
        TextField{
            id: txt_py_address
            Layout.fillWidth: true
            placeholderText: "python main address"
            selectByMouse: true

            text: "D:/projects/action_detection/bachelor-
project/venv/Scripts/python.exe D:/projects/action_detection/bachelor-
project/main.py"
        }

        Button{
            text: "set"

            onClicked:{
                py.setPythonMainScript(txt_py_address.text)
                popup.close()
            }
        }
    }
}
```

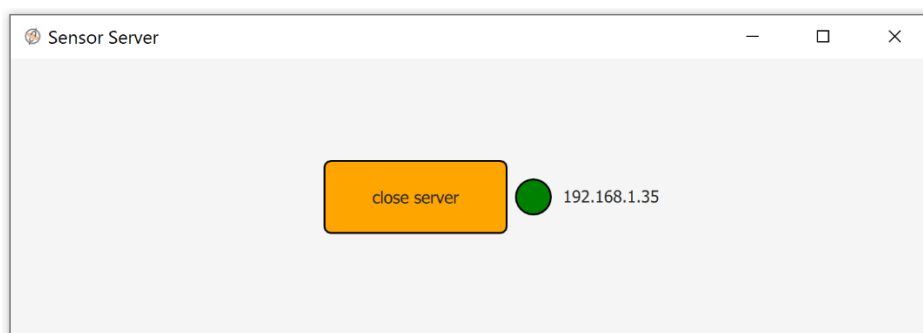
در نهایت نیز خروجی برنامه بصورت زیر می باشد.



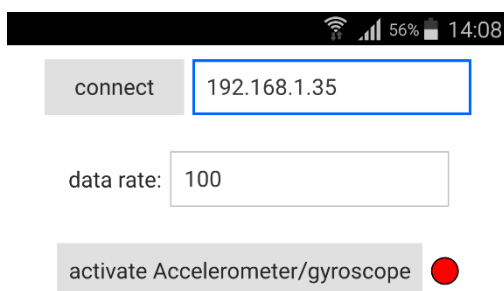
## فصل پنجم

### نحوه جمع آوری مجموعه داده

در ابتدا درباره ی جمع آوری مجموعه داده و مراحل آن توضیح داده خواهد شد.  
ابتدا برنامه دسکتاپ را اجرا کرده و سرور را برای وصل شدن موبایل به آن باز می کنیم.



سپس برنامه موبایل را که توسط آقای اسدی توسعه داده شده است را اجرا کرده و آی پی نمایش داده شده در برنامه دسکتاپ را وارد می کنیم.





سپس اتصال را برقرار کرده و سنسورهای شتاب سنج وژیروسکوپ را نیز فعال می کنیم تا اطلاعات این حسگر ها را ذخیره کنیم.

14:0855%

disconnect192.168.1.35

data rate:100

deactivate Accelerometer/gyroscope

connecting to 192.168.1.35...  
connected

-0.3106476664543152  
1.4395138025283813  
12.29781723022461

7.141113758087158  
-28.503419876098633  
-23.13232421875

و هنگامی که حرکت را انجام دادیم، بلافاصله دکمه disconnect را می زنیم تا ارتباط قطع شود.

14:0855%

connect192.168.1.35

data rate:100


activate Accelerometer/gyroscope

connecting to 192.168.1.35...  
connected  
disConnect

0.166995570063591  
1.2761094570159912  
9.409811019897461

9.70458984375  
-5.126953601837158  
-1.8310546875

پس از این فرآیند، فایل CSV مربوط به حرکت، در پوشه ای که برنامه دسکتاپ قرار دارد ساخته می شود.

 2020\_04\_23-22\_04\_53\_500.csv

8/25/2020 9:29 PM

Microsoft Excel C...

67 KB

و محتویات درون به صورت زیر می باشد.

```
ACC_or_GYR,millisecond,x,y,z
ACC,232,9.210000038146973,0.4399999976158142,1.5999999046325684
GYR,233,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,234,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,235,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,235,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,235,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,235,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,235,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,236,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,236,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,236,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,236,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,236,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,236,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,237,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,237,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,237,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,237,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,239,9.130000114440918,0.3499999940395355,1.5199999809265137
GYR,239,-1.718873381614685,3.43774676322937,-1.718873381614685
ACC,246,9.149999618530273,0.3799999952316284,1.5499999523162842
GYR,248,0,2.2918312549591064,-0.5729578137397766
ACC,249,9.149999618530273,0.4099999964237213,1.5199999809265137
GYR,249,0,2.2918312549591064,-0.5729578137397766
ACC,266,9.170000076293945,0.4699999988079071,1.4900000095367432
GYR,268,1.1459156274795532,0,0.5729578137397766
```

## نتایج و تحلیل آنها

در بخش یادگیری ماشین، پس از جداکردن داده های مجموعه داده به ۲ بخش آموزش و تست، داده های آموزش را به عنوان ورودی به الگوریتم SVM داده تا به یادگیری بپردازد. بعد از این بخش، با داده های تست، الگوریتممان را تست کرده و به درصد درستی ۹۶.۵٪ رسیده ایم. این درصد با کمک ۱۰ ویژگی بدست آمد. بدلیل اینکه اندازه مجموعه داده، معقول بوده و سرعت پاسخ الگوریتم مناسب است، نیازی به کاهش ویژگی ها نداشتیم. اگرچه برای آزمایش، ۲ ویژگی کشیدگی و چولگی را حذف نموده و به درصد درستی ۹۵.۱٪ رسید و دوباره این ۲ ویژگی را برگرداندیم.

## فصل ششم

### خلاصه

اگر بطور خلاصه بخواهیم درباره این پروژه صحبت کنیم. این پروژه به ۳ بخش کلی یادگیری ماشین، برنامه موبایل و برنامه دسکتاپ می باشد که در بخش یادگیری ماشین، عمده کار بخش الگوریتم آن و بخش برنامه دسکتاپ بر عهده اینجانب بود.

ابتدا به جمع آوری مجموعه داده پرداخته و ۷۰۰ داده جمع آوری شد. در بخش الگوریتم انتخابی برای یادگیری ماشین، پس از بررسی و آزمایش الگوریتم های مختلف توسط مجموعه داده ای که فراهم آوردیم، به این نتیجه رسیدیم که الگوریتم SVM مناسب ترین الگوریتم برای این پروژه می باشد و این الگوریتم انتخاب شد. ویژگی های میانگین، میانه، انحراف معیار، واریانس، مینیمم، ماکزیمم، چارک اول، چارک سوم، کشیدگی، چولگی و دامنه بین چارکی نیز به عنوان ویژگی های قابل استخراج انتخاب شد که به درصد درستی ۹۶.۵٪ ختم شد که نتیجه قابل قبولی می باشد.

در بخش برنامه دسکتاپ نیز یک برنامه برای ساخت مجموعه داده و یک برنامه برای تشخیص حرکات ورودی و اعلام آن پیاده سازی شد.

### نقاط قوت پروژه

در بخش ماشین لرنینگ این پروژه، درصد درستی ۹۶.۵٪، نقطه قوت می باشد که حرکات را به خوبی تشخیص می دهد. در بخش برنامه دسکتاپ نیز چون پایه برنامه نوشته شده، زبان C++ می باشد که یکی از سریعترین زبان های برنامه نویسی است، برنامه از سرعت خوبی برخوردار است و اگر مقیاس این پروژه در آینده افزایش یابد، به مشکلی از لحاظ سرعت برنخواهد خورد.

### نقاط ضعف پروژه

در این پروژه، ۴ حرکت تشخیص داده می شود و برای تشخیص حرکت نیز باید حرکات به صورت مجزا انجام شوند و نمی توان دنباله ای از حرکات را انجام داد. می توان به این ۲ مورد به عنوان نقاط ضعف نگاه کرد.

## نتیجه گیری

تلفن های همراه دستگاه های بسیار به روز و قدرتمندی هستند که علاوه بر کارهای روزمره، کارهای بسیار زیاد دیگری را با آنها می توان انجام داد. حسگرهای تلفن های همراه بسیار کارآمد هستند و با کمک ۲ حسگرژیروسکوپ و شتاب سنج، قادر خواهیم بود که حرکات افراد را تشخیص دهیم. استفاده از این سنسورها و دادن اطلاعات آنها به الگوریتم های یادگیری ماشین و یادگیری عمیق، می توان به نتایج بسیار مفیدی رسید و قطعاً در آینده نیز استفاده های بسیار زیادی از آن خواهد شد.

## کارهای آینده

در این پروژه، ۴ حرکت تشخیص داده شد و در آینده، حرکات جدیدتر اضافه خواهد شد و در عین حال سعی خواهیم داشت که درصد درستی تغییر زیادی نکند. همچنین در حال حاضر، حرکات به صورت مجزا تشخیص داده می شوند و در آینده، تشخیص دنباله ای از حرکات را نیز اضافه خواهیم کرد.

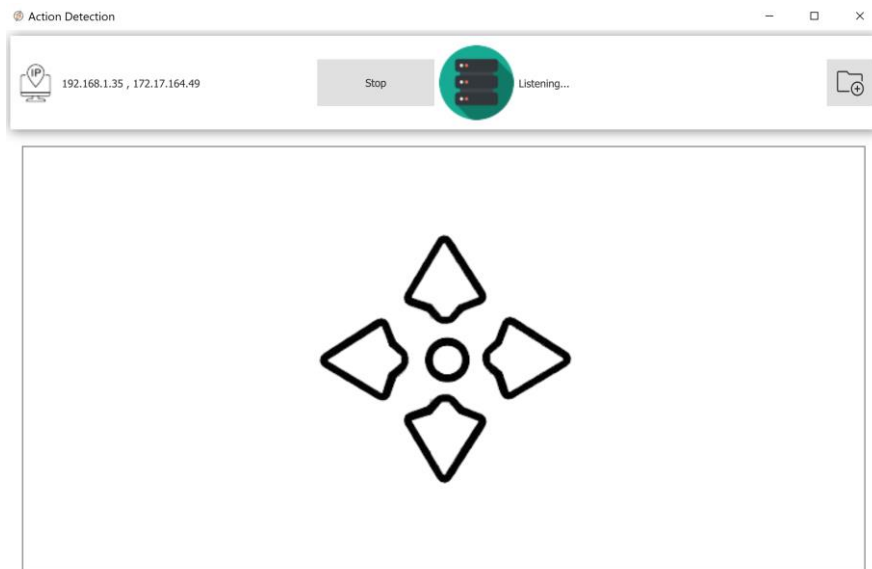
## مراجع

- [1] P. K. Roy, H. Om, Suspicious and violent activity detection of humans using hog features and svm classifier in surveillance videos, in: *Advances in Soft Computing and Machine Learning in Image Processing*, Springer, 2018, pp. 277–294
- [2] A. Thyagarajmurthy, M. Ninad, B. Rakesh, S. Niranjana, B. Manvi, Anomaly detection in surveillance video using pose estimation, in: *Emerging Research in Electronics, Computer Science and Technology*, Springer, 2019, pp. 753–766.
- [3] M. Ehatisham-Ul-Haq, A. Javed, M. A. Azam, H. M. Malik, A. Irtaza, I. H. Lee, M. T. Mahmood, Robust human activity recognition using multimodal feature-level fusion, *IEEE Access* 7 (2019) 60736–60751.
- [4] E. Zdravevski, P. Lameski, V. Trajkovic, A. Kulakov, I. Chorbev, R. Goleva, N. Pombo, N. Garcia, Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering, *IEEE Access* 5 (2017) 5262–5280.
- [5] T. Billah, S. M. Rahman, M. O. Ahmad, M. Swamy, Recognizing distractions for assistive driving by tracking body parts, *IEEE Transactions on Circuits and Systems for Video Technology* 29 (4) (2018) 1048–1062.
- [6] L. Martínez-Villaseñor, H. Ponce, A concise review on sensor signal acquisition and transformation applied to human activity recognition and human–robot interaction, *International Journal of Distributed Sensor Networks* 15 (6) (2019) 1550147719853987.
- [7] R. Mojarad, F. Attal, A. Chibani, S. R. Fiorini, Y. Amirat, Hybrid approach for human activity recognition by ubiquitous robots, in: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 5660–5665.
- [8] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz<sup>1</sup>, “A Public Domain Dataset for Human Activity Recognition Using Smartphone’s,” *ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges (Belgium), 24-26 April 2013.
- [9] Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore, “Activity Recognition using Cell Phone Accelerometers,” *ACM SIGKDD Explorations Newsletter* Volume 12 Issue 2, Pages 74-82, December 2010.
- [10] B. Jagadeesh, C. M. Patil, Video based human activity detection, recognition and classification of actions using svm, *Transactions on Machine Learning and Artificial Intelligence* 6 (6) (2019) 22.

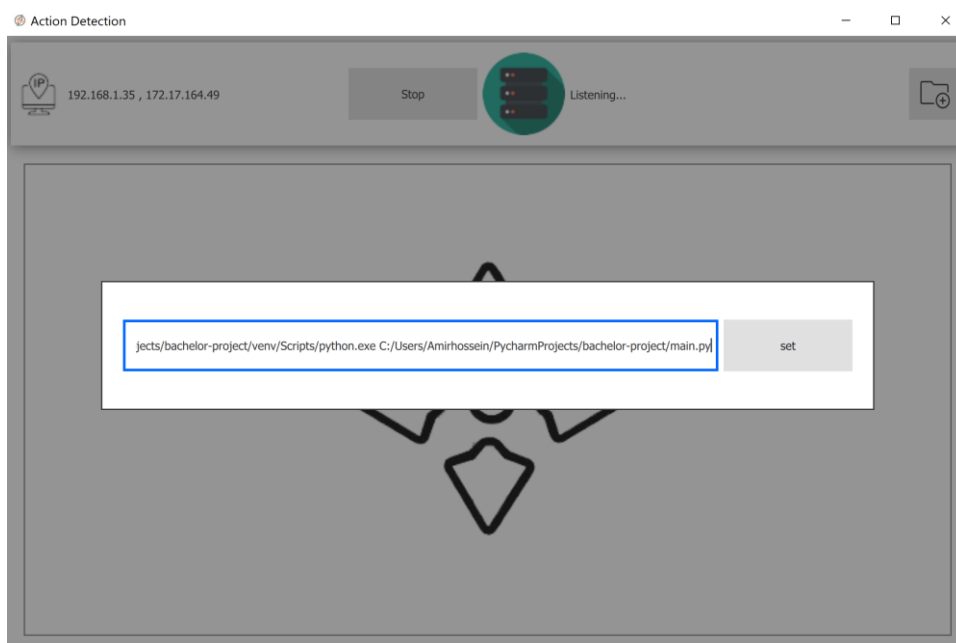
## پیوست

### نحوه کار با برنامه

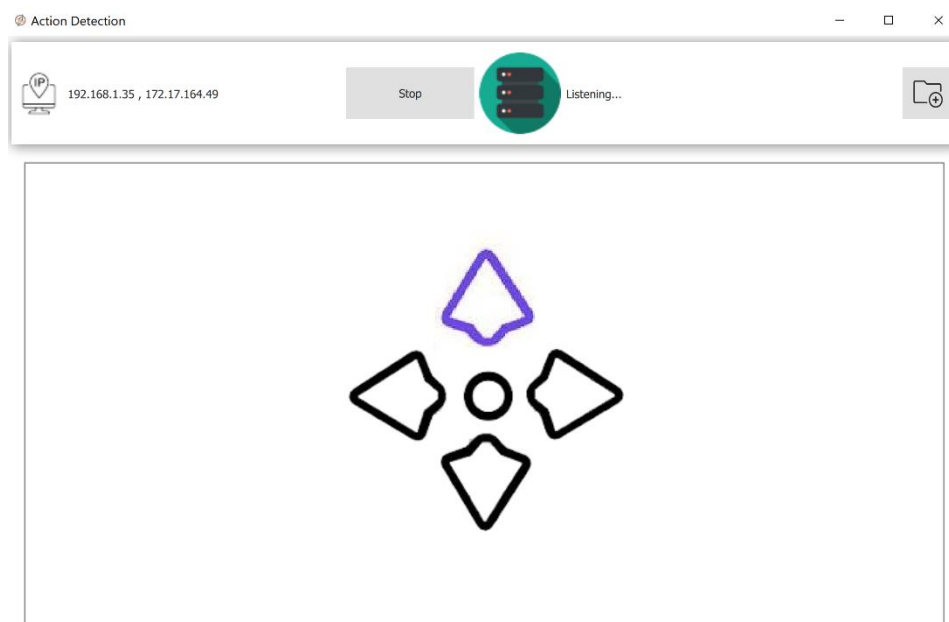
ابتدا برنامه دومی که برای دسکتاپ زده شده را باز می کنیم و سرور را روشن می کنیم.



سپس آدرس فایل `main.py` را وارد می کنیم.



با همان برنامه موبایل که قبلتر معرفی شد، به دسکتاپ متصل می شوید و حرکت موردنظر را انجام داده و نتیجه بصورت زیر مشخص می شود.



\*نحوه حرکات دست، در پوشه **videos** قرار دارد.



**University of Tehran  
College of Farabi  
Faculty of Engineering  
Department of Computer Engineering**

# **Desktop Application Development for an Activity Recognition System Based on Mobile Gyroscope Sensor**

**By:**

**Amirhossein Barari**

**Under Supervision of:**

**Dr. Kazim Fouladi**

**A Project Report as a Requirement for  
the Degree of Bachelor of Science in  
Computer Engineering**

**September 2020**



