



Вградена апликација за микроконтролер
за препознавање на
лица, звук и облици во склоп на
биолошки роботски систем

Софтвер за вградливи системи

Проф. д-р Моника Симјаноска

Зорица Коцева

Содржина:

- Идеја на системот
- Што опфаќа системот?
- Хардвер
- Софтвер
- Користени елементи во Felgo
- МУ на слики, објекти и предмети во Felgo
- [Линк до код](#)



Идеја на проектот

- Детекција на предмети, памтење и препознавање на лица, користење на сензори и посебни страни за користење на уредот и за приватни потреби при поставување во робот, користење во индустрија, здравствени установи и слично.
- Дополнително мултумодална фузија со имплементација за говор и звук-текст може да се додаде

Што е замислено да опфаќа системот?


- Дисплеј и микроконтролер кој ќе биде најпогоден за ваквиот тип на систем прво како прототип на симулатор, по можност потоа да се развие на вистински уред. Софтверот треба да соодветствува на функциите кои ги има хардверот, со што самата апликација ќе вклучува камера, процесирање на слика и звук од лице, по можност и враќање на асистивен одговор. Искористен е опсегот на повеќе извршувачки процеси- таскови и нишки за самата апликација во реално време според FreeRTOS.



Хардвер



Концепт модел на хардверот

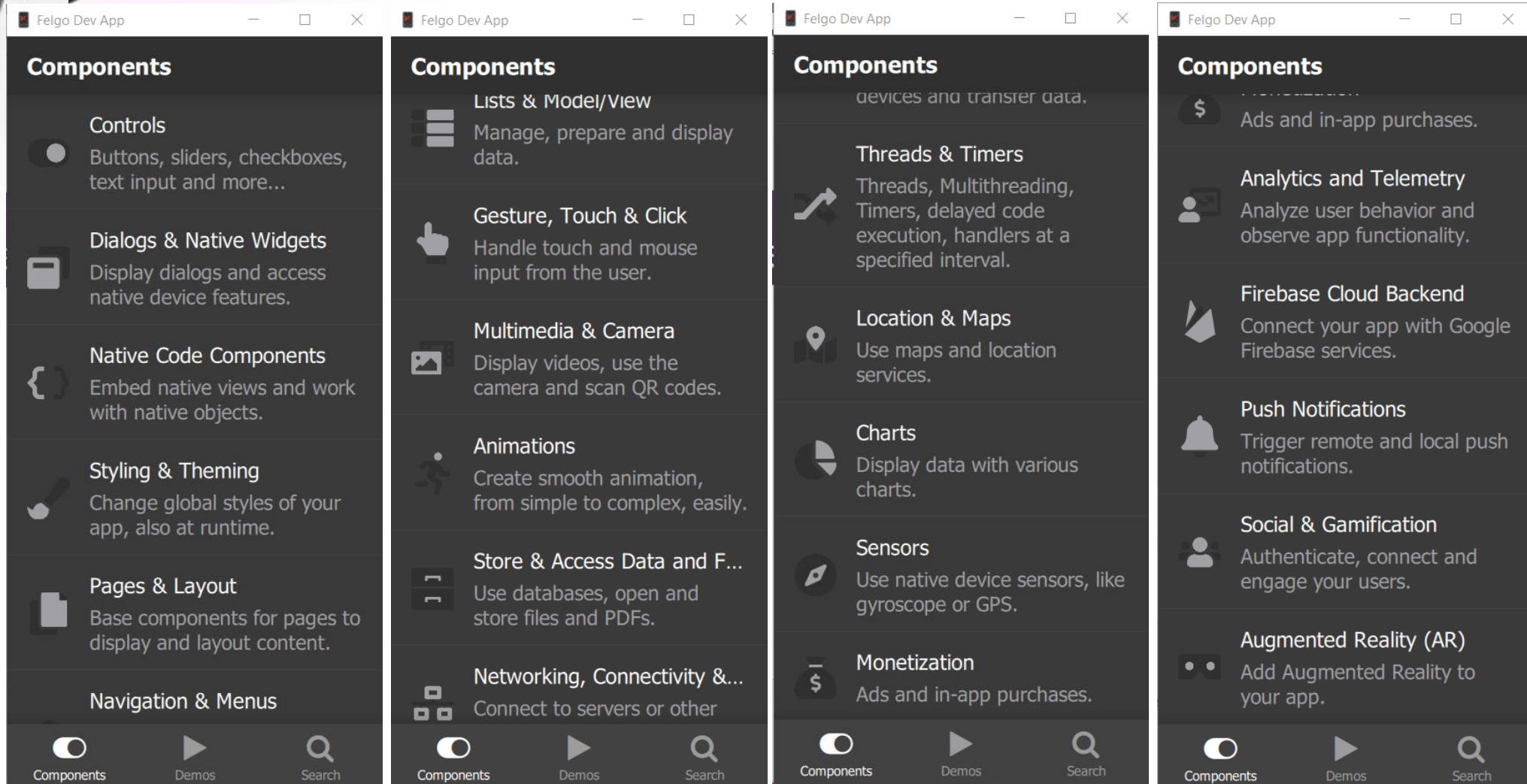


Машинско учење: Класификација на слики со Qt и TensorFlow

- Вештачката интелигенција и паметните апликации постојано стануваат се попопуларни. Компаниите силно се потпираат на системите за вештачка интелигенција и машинското учење за да донесат побрзи и попрецизни одлуки врз основа на нивните податоци. Ова дава пример за класификација на слики и откривање објекти изградени со TensorFlow Framework на Google.

Користени елементи во Felgo

- Demos-Examples

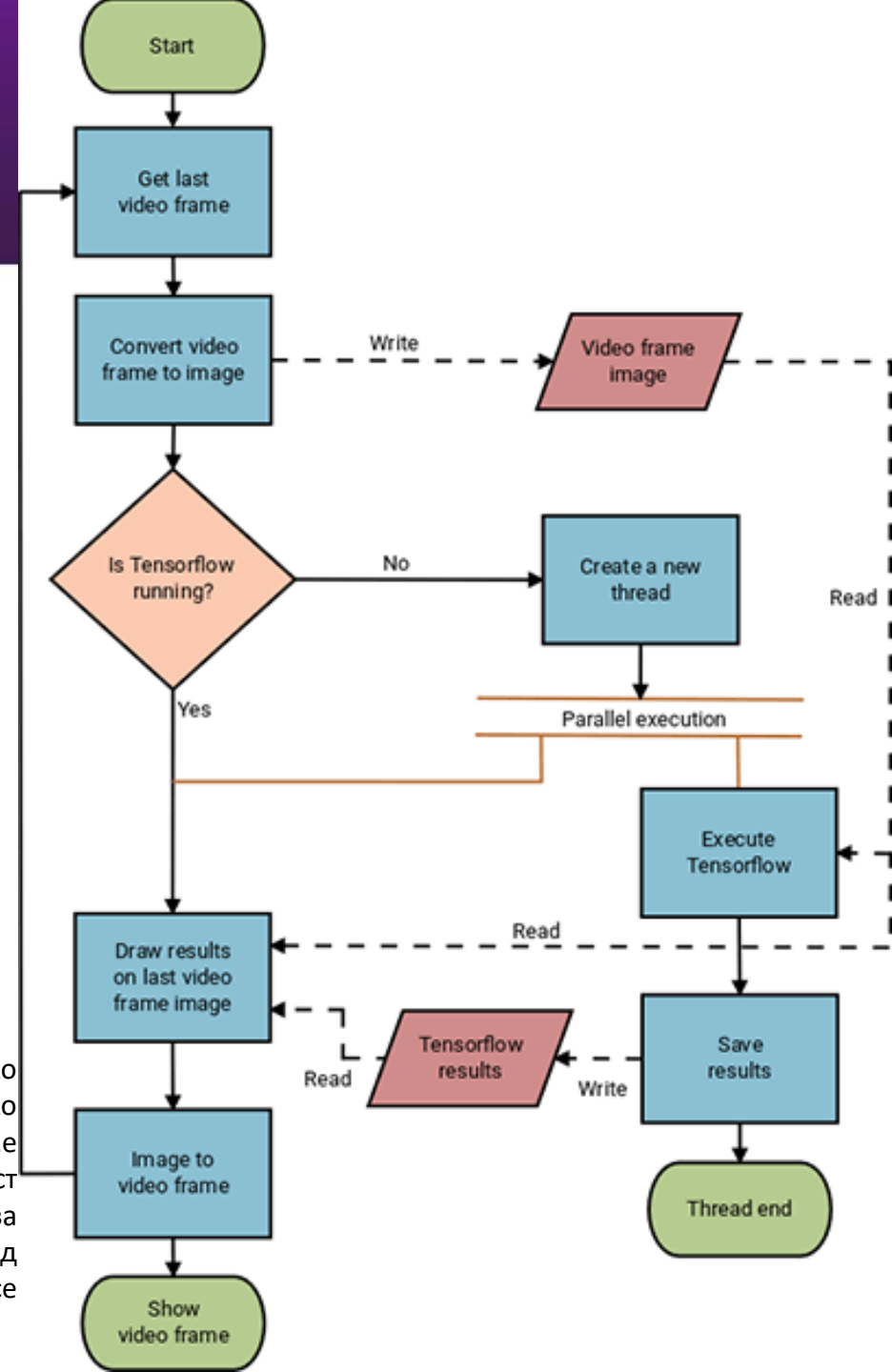


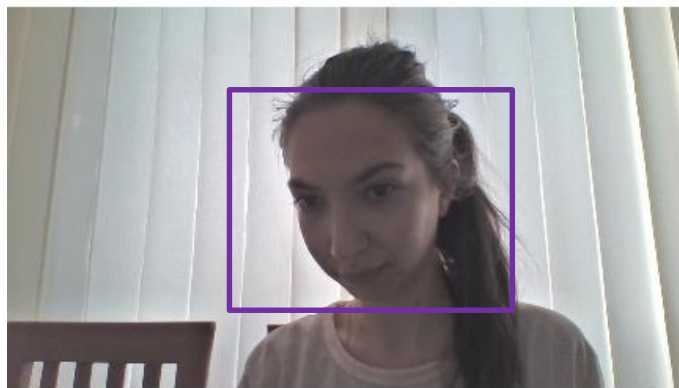
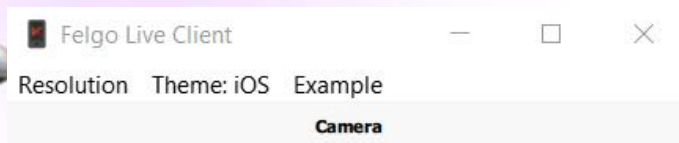
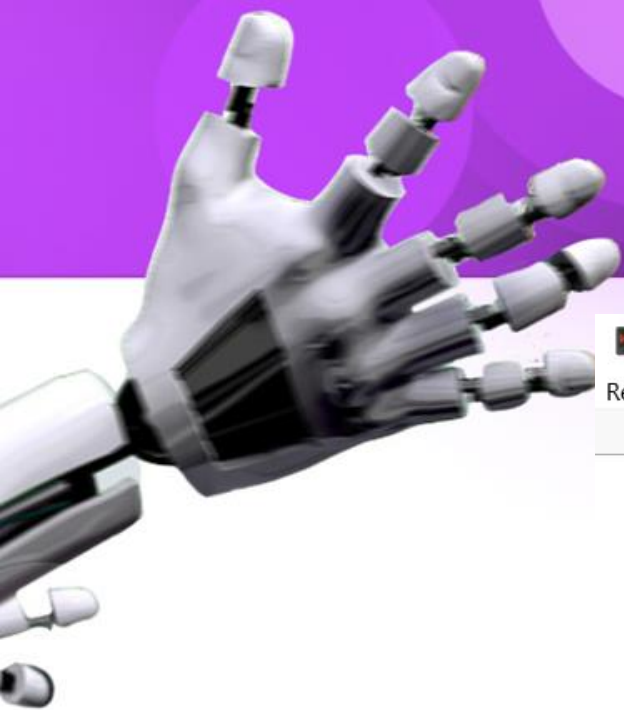
Како работи со слики и видеа?

Филтерот се обработува во методот `run` од класата `ObjectsRecogFilter`. Општите чекори се следните.

1. Треба да го конвертираме нашиот `QVideoFrame` во `QImage` за да можеме да манипулираме со него.
2. Проверуваме дали работи Tensorflow. Бидејќи Tensorflow се извршува во друга нишка, ги користевме класите `QMutex` и `QMutexLocker` за да провериме дали нишката работи. Убав пример е даден во документацијата `QMutexLocker Class`. Ако Tensorflow работи - ништо не е направено. Ако Tensorflow НЕ работи - го извршуваме во друга нишка со помош на класите C++: `TensorflowThread` и `WorkerTF`, сигналите и слотот се користат за комуникација на главната нишка и овие класи, проверете `[QThreads general usage]` (https://wiki.qt.io/QThreads_general_usage) за дополнителни детали. Како влез ја обезбедуваме сликата на видео рамката. Кога Tensorflow е завршен, ги складираме резултатите дадени за избраниот модел, исто така, со помош на сигнали и слотови.

3. Ги добиваме зачуваните резултати (ако ги има) и ги применуваме на тековната слика на видео рамката. Ако нашиот модел е класификација на слики, ние само ги цртаме името и резултатот на врвната класа на слики ако резултатот е над минималната вредност на доверба. Ако нашиот модел е детекција на објекти, ги повторуваме сите детекции и ги цртаме полињата за ограничување, имињата на објектите и вредностите на доверливост доколку се над минималното ниво на доверба. Постои дополнителна класа C++, `AuxUtils`, која обезбедува функции за цртање на рамки, како што се `drawText` и `drawBoxes`. 4. Последниот чекор е да го претвориме назад нашиот `QImage` во `QVideoFrame` што ќе биде обработен од нашата компонента `QML VideoOutput` и потоа се враќаме да обработиме нова видео рамка.





```
Item {
    width: 640
    height: 360

    Camera {
        id: camera

        imageProcessing.whiteBalanceMode: CameraImageProcessing.WhiteBalanceFlash

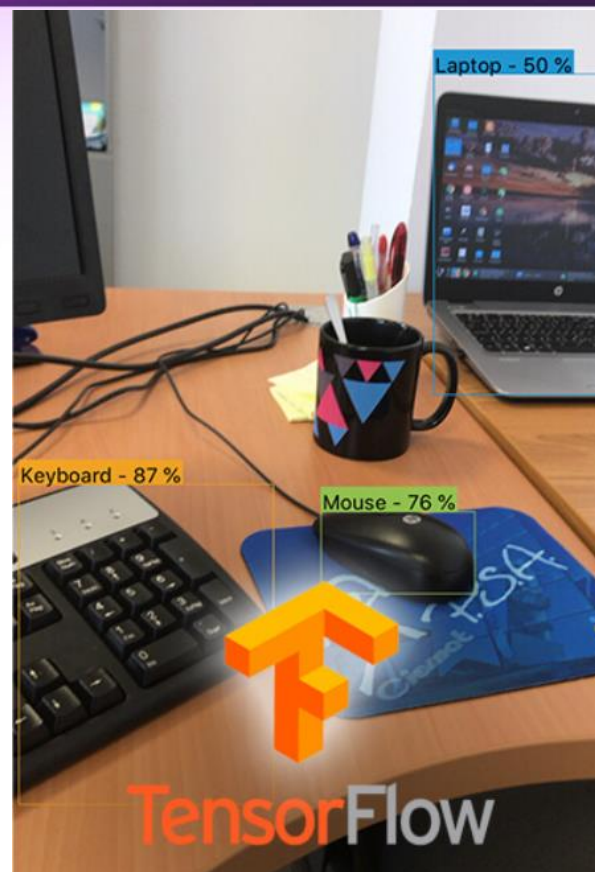
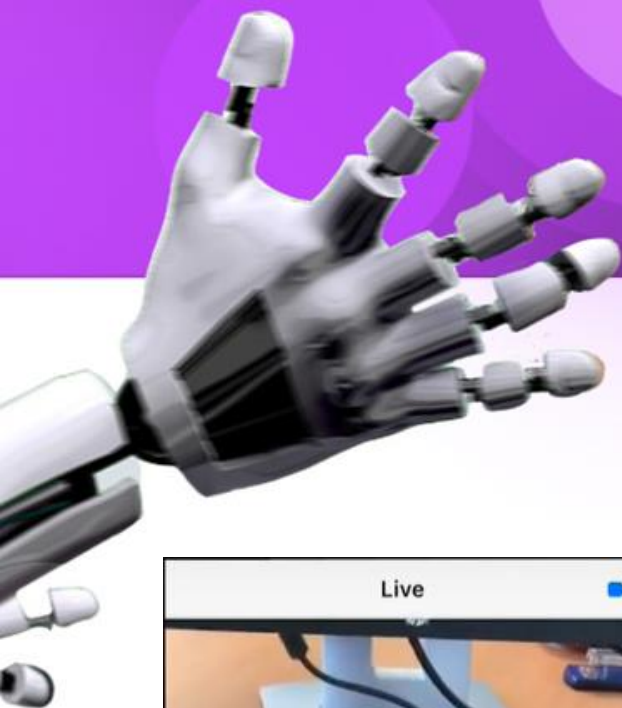
        exposure {
            exposureCompensation: -1.0
            exposureMode: Camera.ExposurePortrait
        }

        flash.mode: Camera.FlashRedEyeReduction

        imageCapture {
            onImageCaptured: {
                photoPreview.source = preview // Show the preview in an Image
            }
        }
    }

    VideoOutput {
        source: camera
        anchors.fill: parent
        focus : visible // to receive focus and capture key events when visible
    }

    Image {
        id: photoPreview
    }
}
```



Settings

Minimum confidence



Show inference time



Tensorflow model

☐ Image classification

☒ Object detection



```
VideoOutput {  
    id: videoOutput  
    anchors.fill: parent  
    source: camera  
    visible: camera.availableCamera && camera.cameraStatus ==  
Camera.ActiveStatus  
    autoOrientation: true  
    fillMode: VideoOutput.PreserveAspectCrop  
    rotation: initialRotation()  
  
    filters: [objectsRecognitionFilter]  
}
```

- Видео излезот од камерата ја исполнува целата страница. Тоа е видно само кога барем една камера е откриена и активна.
- Ние дефинираме филтер `objectsRecognitionFilter` кој се имплементира во класа C++. Овој филтер ја добива секоја видео рамка, ги трансформира како влезни податоци во TensorFlow, го повикува TensorFlow и ги црта резултатите преку видео рамката.



Позитивни страни

- Qt има богат сет на готови за употреба мултиплатформски компоненти за различни области како што се мултимедија, мрежа и поврзување, графика, методи на внесување, сензори, складирање податоци и друго. Felgo дополнително придонесува за олеснување на распоредувањето на мобилни и вградени уреди и додава убави функции како што се независност на резолуцијата и соодносот и дополнителни компоненти и контроли, обезбедува полесен пристап до природните функции, како и приклучоци за монетизација, аналитика, облак услуги и многу повеќе.
- Една убава карактеристика на Felgo е тоа што не е ограничена на мобилни уреди, така што можете да ја тестирате и прототипирате вашата апликација во вашиот компјутер за развој, што секако е побрзо од компајлирањето и распоредувањето на вашата апликација на емулатори. Можете дури и да користите повторно вчитување Felgo во живо за да ги видите промените во кодот речиси моментално. Повторното вчитување во живо е поддржано и на уредите со Android и iOS, што е совршено за дотерување промени или тестирање фрагменти од код на мобилни уреди. Така, Tensorflow ја обезбедува рамката за машинско учење, додека Felgo и Qt го олеснуваат распоредувањето на апликацијата на повеќе платформи: десктоп и мобилни.