Kotlin & Android w Android Studio (3.5.1)

Rozpoznawanie mowy. Latarka. Czujnik natężenia światła.

Tym razem utworzymy aplikację demonstrującą wykorzystanie API Google do rozpoznawania mowy oraz uruchamianie diody aparatu jako latarki. Zajmiemy się również czujnikiem natężenia światła.

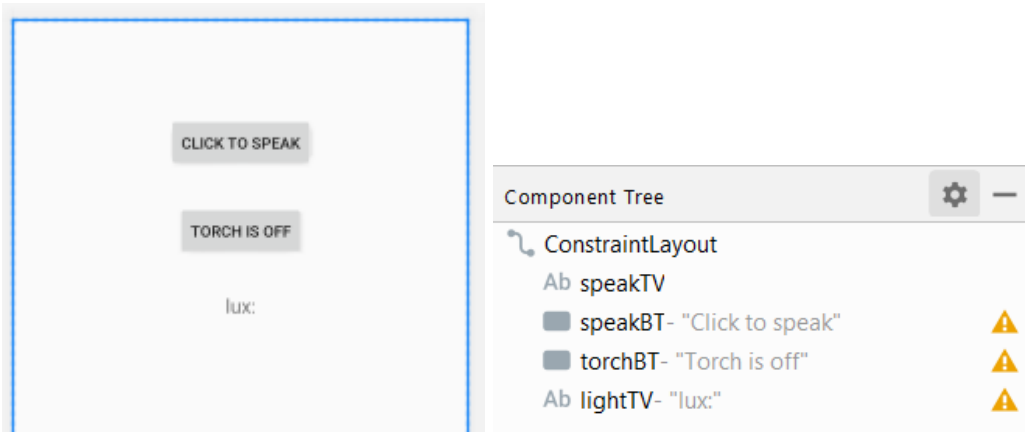
Idea pierwszej części naszej aplikacji polegać będzie na możliwości uruchomienia rozpoznawania mowy po kliknięciu przycisku. Efektem działania będzie wyświetlenie na ekranie wypowiedzianego tekstu. Ponadto kliknięcie na kolejny przycisk umożliwiać będzie uruchomienie bądź wyłączenie latarki (diody doświetlającej aparatu). Ostatnią częścią programu będzie wyświetlanie poziomu natężenia światła w luksach.

Interfejs aplikacji

Przygotujmy layout naszej aplikacji.

ZADANIE

Utwórz nowy projekt Android Studio w języku Kotlin o nazwie **VoiceRec_Torch_LightInt**, zawierający **EmptyActivity**. W pliku layoutu głównej aktywności użyj komponentu **ConstraintLayout** (domyślnie ustawiony). Dodaj do layoutu komponenty:
- **TextView** (id=speakTV, marginTop=32dp, text=””),
- poniżej 2x **Button** (id=speakBT, marginTop=32dp, text=”Click to speak” oraz id=torchBT, marginTop=32dp, text=”Torch is off”)
- niżej kolejny **TextView** (id=lightTV, marginTop=32dp, text=”lux: ”).



Layout naszej aplikacji jest gotowy, przystępujemy do oprogramowania funkcji.

Funkcjonalności aplikacji

Na początek dodajmy w pliku manifestu stosowne uprawnienia dla naszej aplikacji, tj. możliwość nagrywania dźwięku, dostęp do aparatu oraz dostęp do czujnika natężenia światła.

ZADANIE

Dodaj w pliku manifestu poniższe uprawnienia.

```
<uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
<uses-permission android:name="android.permission.CAMERA"></uses-permission>
<uses-feature android:name="android.hardware.sensor.light"
    android:required="true"></uses-feature>
```

Rozpoznawanie mowy

Przystępujemy do oprogramowania pierwszej funkcjonalności – rozpoznawanie mowy.

ZADANIE

Zaimplementuj poniższy kod przycisku w metodzie **onCreate()**.

```
speakBT.setOnClickListener { it: View!
    val intent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())
    startActivityForResult(intent, requestCode: 10)
}
```

Po kliknięciu na przycisk „Click to speak” wywołana zostanie intencja rozpoznawania mowy.

W pierwszej linii tworzymy intencję rozpoznawania mowy. Dalej informujemy intencję, który model mowy będzie preferowany podczas wykonywania ACTION_RECOGNIZE_SPEECH (wymagane). Program rozpoznający mowę wykorzystuje te informacje do dokładnego dostrojenia wyników. W naszym przypadku używamy modelu językowego opartego na swobodnym rozpoznawaniu mowy (LANGUAGE_MODEL_FREE_FORM).

Następnie określamy język, w którym wypowiadane będą słowa. Ustawiamy tutaj domyślny język telefonu (Locale.getDefault()). Możemy również wybrać jeden z dostępnych języków, np.:

```
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.ENGLISH)
```

Na końcu uruchamiamy intencję wykorzystując metodę **startActivityForResult()**, a w dalszej części kodu obsłużymy wynik działania intencji za pomocą funkcji **onActivityResult()**.

ZADANIE

Dodaj poniższą funkcję w klasie **MainActivity**.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (resultCode == Activity.RESULT_OK && data != null) {
        if (requestCode==10) {
            val stringArrayListExtra : ArrayList<String!>! = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS)
            speakTV.setText(stringArrayListExtra[0])
        }
    } else {
        Toast.makeText(applicationContext, text: "Failed to recognize speech!", Toast.LENGTH_LONG).show()
    }
}
```

Obsługa wyniku działania intencji do rozpoznawania mowy sprowadza się do pobrania danych **EXTRA** i wyświetleniu zwróconego przez intencję tekstu w polu **TextView**.

Latarka

Teraz zajmiemy się drugą z funkcjonalności – latarką. Funkcjonalność latarki zapewni nam dioda doświetlająca tylnego aparatu telefonu. Przystępujemy do implementacji.

ZADANIE

Zaimplementuj poniższy kod w metodzie **onCreate()** głównej aktywności.

```
var torchOn = false
val cameraManager : CameraManager  = getSystemService(Context.CAMERA_SERVICE) as CameraManager
val cameraId : String!  = cameraManager.cameraIdList[0]
torchBT.setOnClickListener { it: View!
    if(!torchOn)
    {
        cameraManager.setTorchMode(cameraId,  enabled: true)
        torchOn = true
        torchBT.setText("Torch is on")
    }
    else{
        cameraManager.setTorchMode(cameraId,  enabled: false)
        torchOn = false
        torchBT.setText("Torch is off")
    }
}
```

Ponieważ w naszym layoucie umieściliśmy pojedynczy przycisk do obsługi latarki, dlatego musimy użyć dodatkowej zmiennej logicznej do reprezentowania aktualnego stanu latarki. W powyższym kodzie zmienna **torchOn** przyjmując wartość **false** oznacza wyłączoną latarkę, zaś **true** – latarkę włączoną. Wartość zmiennej aktualizujemy poprzez naciśnięcie przycisku w następujący sposób: jeśli **torchOn** ma wartość **false** to przy naciśnięciu przycisku zmieniamy ją na **true** i odwrotnie.

W linii drugiej tworzymy uchwyt do usługi aparatu (CAMERA_SERVICE) z poziomu systemu Android. Następnie pozyskujemy identyfikator aparatu korzystając z listy identyfikatorów wszystkich aparatów dostępnych w naszym telefonie. Identyfikator tylnego aparatu (głównego) znajduje się na indeksie 0.

Mając dostęp do tylnego aparatu możemy uruchamiać jego diody w trybie latarki wykorzystując metodę **setTorchMode()** wywoływaną na managerze aparatów. I tak w obsłudze akcji przycisku latarki, jeśli **torchOn**=**false** to włączamy latarkę, **torchOn** ustawiamy na **true** oraz zmieniamy tekst na przycisku („Torch is on”). Przeciwnie czynimy w przypadku kliknięcia na przycisk latarki, gdy jest już ona uruchomiona (gdy **torch**=**true**).

Czujnik natężenia światła

Czujnik natężenia światła umożliwia pomiar oświetlenia otoczenia. W systemie Android wykorzystywany jest przede wszystkim do dostosowania jasności ekranu czy też zmiany wyglądu aplikacji (poprawa widoczności, kontrast).

W naszej aplikacji sensor ten wykorzystamy jedynie do pomiaru natężenia światła. Efektem będzie wyświetlenie uzyskanej wartości w stosunku do wartości maksymalnej.

Zaczynamy od utworzenia niezbędnych zmiennych i uchwytu do czujnika.

ZADANIE

Dodaj w **MainActivity** zmienne:

```
class MainActivity : AppCompatActivity() {

    lateinit var sensorManager: SensorManager
    lateinit var lightSensor: Sensor
```

Następnie w metodzie **onCreate()** zainicjalizuj managera czujników oraz uchwyt do czujnika światła.

```
sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
lightSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
```

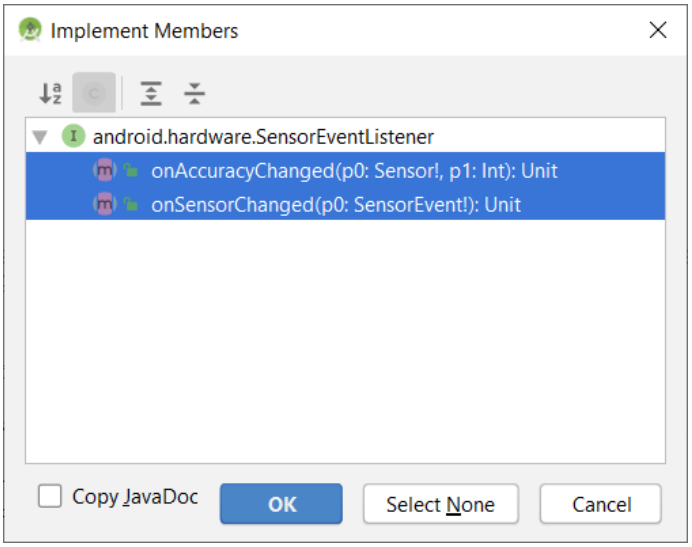
Aby na bieżąco otrzymywać aktualne dane z sensora nasza klasa musi implementować interfejs **SensorEventListener**. Musimy również dodać odpowiednie metody „reagujące” na zmianę wskazań czujnika.

ZADANIE

Dodaj do nagłówka **MainActivity** interfejs **SensorEventListener**:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
```

Następnie wewnątrz klasy dodaj sugeowane przez kompilator funkcje:




```
override fun onAccuracyChanged(p0: Sensor?, p1: Int) {
}

override fun onSensorChanged(event: SensorEvent) {
}
```

Pierwsza z funkcji wywoływana jest automatycznie, gdy zmieniła się dokładność czujnika, zaś druga w przypadku zmiany wartości mierzonej przez sensor. Nas interesują druga z metod.

ZADANIE

Dodaj do funkcji **onSensorChanged()** poniższy kod:

```
override fun onSensorChanged(event: SensorEvent) {
    if(event.sensor.type == Sensor.TYPE_LIGHT){
        lightTV.setText("lux: " + event.values[0] +" / " + lightSensor.maximumRange)
    }
}
```

W zaimplementowanej metodzie sprawdzamy czy zdarzenie na sensorze dotyczy czujnika światła. Jeśli tak, to pobieramy aktualną wartość natężenia światła (event.value[0]) oraz maksymalną wartość wskazania dla tego sensora i wyświetlamy ich stosunek używając komponentu **TextView** naszej aplikacji.

To jednak nie wszystko. Musimy jeszcze zarejestrować nasz listener do dokonywania próbkowania sygnału czujnika z określoną częstotliwością.

ZADANIE

Dodaj metodę **onResume()** i zaimplementuj w niej kod:

```
override fun onResume() {
    super.onResume()
    sensorManager.registerListener( listener: this, lightSensor, SensorManager.SENSOR_DELAY_NORMAL)
}
```

Podobnie musimy wyrejestrować nasz listener w momencie zatrzymania działania aplikacji.

ZADANIE

Dodaj metodę **onPause()** i zaimplementuj w niej kod:

```
override fun onPause() {
    super.onPause()
    sensorManager.unregisterListener( listener: this)
}
```