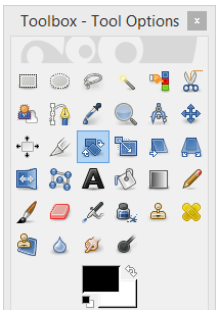
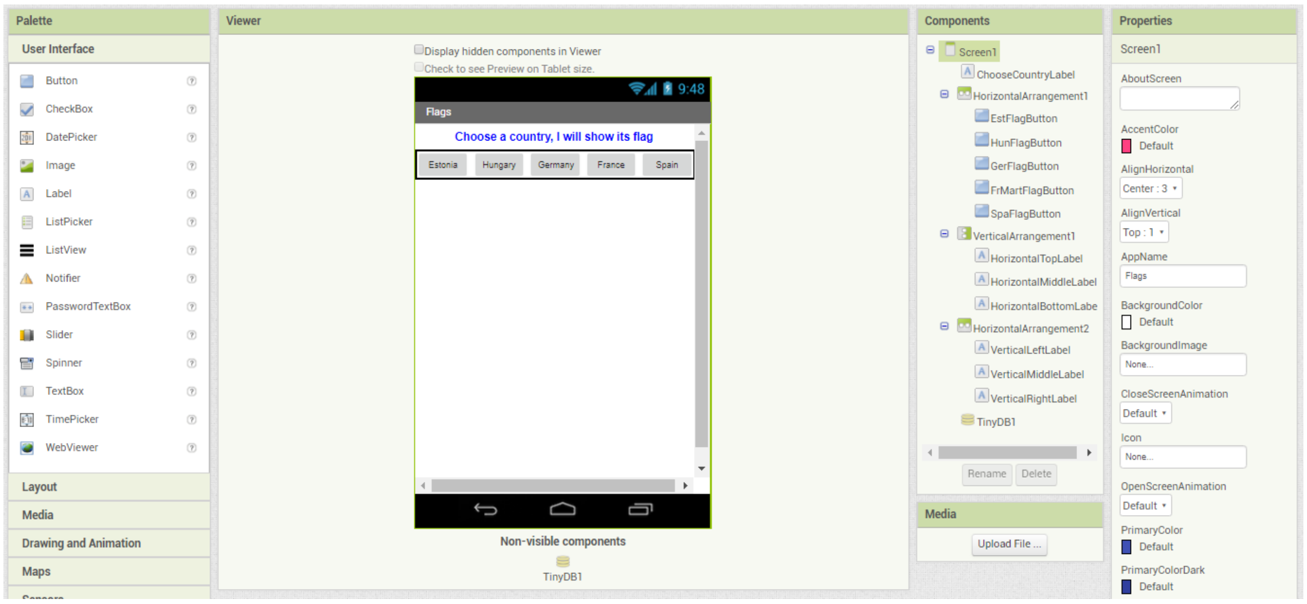
Task 1 | **GIMP** | **Learn the basics**

- Choose an image on the following website: https://apod.nasa.gov/apod/archivepix.html
- Download the chosen image to your computer
- Launch the *GIMP 2* application by clicking on its icon at the desktop
- Try to perform the following modifications to your chosen image:
  - Step 1: *Change the size of the image* then *export the modified image*
  - Step 2: *Crop the image* then *export the modified image*
  - Step 3: *Rotate the image* then *export the modified image*
  - Step 4: *Flip the image* then *export the modified image*
- In the *Toolbox* you can find more, very useful tools to perform several kinds of modifications to any image you choose
- For more tutorials you can visit: https://www.gimp.org/tutorials/



Task 2 | **MIT App Inventor 2** | ***"Flags"***

- Here are the components for the *Flags* app, as shown in the *Component Designer:*



Components

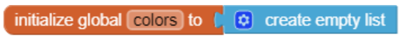
- Screen1
  - ChooseCountryLabel
  - HorizontalArrangement1
    - EstFlagButton
    - HunFlagButton
    - GerFlagButton
    - FrMartFlagButton
    - SpaFlagButton
  - VerticalArrangement1
    - HorizontalTopLabel
    - HorizontalMiddleLabel
    - HorizontalBottomLabe
  - HorizontalArrangement2
    - VerticalLeftLabel
    - VerticalMiddleLabel
    - VerticalRightLabel
  - TinyDB1

Properties

Screen1

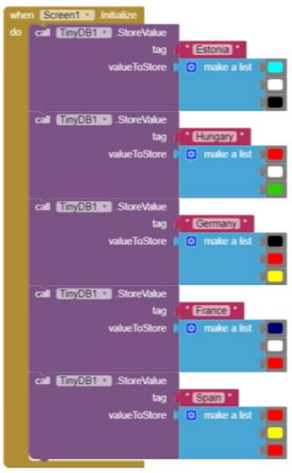
AboutScreen	
AccentColor	Default
AlignHorizontal	Center : 3
AlignVertical	Top : 1
AppName	Flags
BackgroundColor	Default
BackgroundImage	None...
CloseScreenAnimation	Default
Icon	None...
OpenScreenAnimation	Default
PrimaryColor	Default
PrimaryColorDark	Default

- Create a variable, which will be a blank list and name it as '*Color*'.



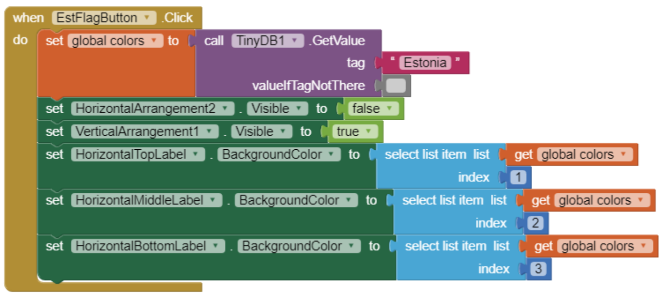
```
initialize global colors to  create empty list
```

- At the start of the application store the colors of the flags in *TinyDB1*. The tag for each flag is the name of the country. ValueToStore has a three-component list where the 3 colors of the flags are stored.
- For example, if we click the button of the flag of Estonia, the values stored in *TinyDB1* under the tag *'Estonia'* will be retrieved by the Get.Value procedure and the variable called *'Color'* will be set to the retrieved value.
  - We turn off the visibility of the *HorizontalArrangement2* and then turn on the visibility of the *VerticalArrangemet1*.



```
when Screen1 .Initialize
do  call TinyDB1 .StoreValue
                 tag  " Estonia "
        valueToStore  make a list  🟦
                                   ⬜
                                   ⬛
    call TinyDB1 .StoreValue
                 tag  " Hungary "
        valueToStore  make a list  🟥
                                   ⬜
                                   🟩
    call TinyDB1 .StoreValue
                 tag  " Germany "
        valueToStore  make a list  ⬛
                                   🟥
                                   🟨
    call TinyDB1 .StoreValue
                 tag  " France "
        valueToStore  make a list  🟦
                                   ⬜
                                   🟥
    call TinyDB1 .StoreValue
                 tag  " Spain "
        valueToStore  make a list  🟥
                                   🟨
                                   🟥
```

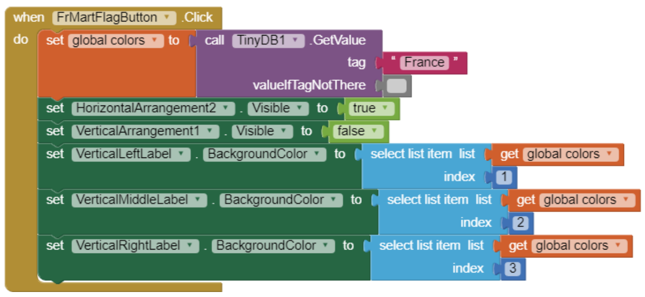
- The background-color of the *HorizontalTopLabel* has to be set to the color stored in the 1st element of the list.
- The background-color of the *HorizontalMiddleLabel* has to be set to the color stored in the 2nd element of the list.
- The background-color of the *HorizontalBottomLabe*l has to be set to the color stored in the 3rd element of the list.



```
when EstFlagButton .Click
do  set global colors to  call TinyDB1 .GetValue
                                tag  " Estonia "
                  valueIfTagNotThere  "  "
    set HorizontalArrangement2 . Visible to  false
    set VerticalArrangement1 . Visible to  true
    set HorizontalTopLabel . BackgroundColor to  select list item  list  get global colors
                                                                  index  1
    set HorizontalMiddleLabel . BackgroundColor to  select list item  list  get global colors
                                                                     index  2
    set HorizontalBottomLabel . BackgroundColor to  select list item  list  get global colors
                                                                     index  3
```

- We do so in every case the flag consists of horizontally arranged stripes by retrieving the data of the appropriate country.

- For example, if we click the button of the flag of France (Martinique), the values stored in *TinyDB1* under the tag *'France'* will be retrieved by the Get.Value procedure and the variable called *'Color'* will be set to the retrieved value.
  - We turn on the visibility of the *HorizontalArrangement2* and then turn off the visibility of the *VerticalArrangemet1*.
  - The background-color of the *HorizontalLeftLabel* has to be set to the color stored in the 1st element of the list.
  - The background-color of the *HorizontalMiddleLabel* has to be set to the color stored in the 2nd element of the list.
  - The background-color of the *HorizontalRightLabe*l has to be set to the color stored in the 3rd element of the list.

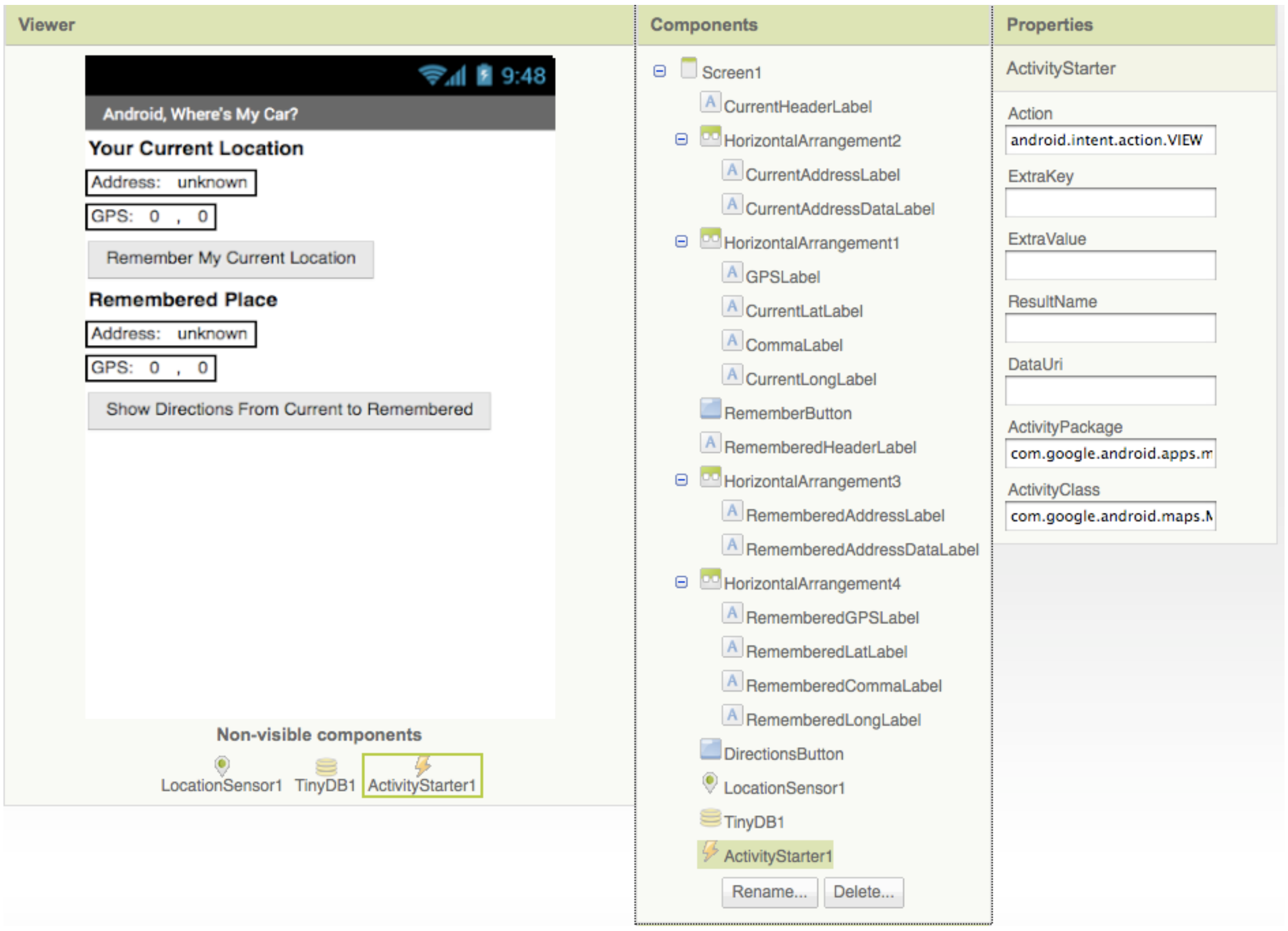


```
when FrMartFlagButton .Click
do  set global colors to  call TinyDB1 .GetValue
                                tag  " France "
                  valueIfTagNotThere  "  "
    set HorizontalArrangement2 . Visible to  true
    set VerticalArrangement1 . Visible to  false
    set VerticalLeftLabel . BackgroundColor to  select list item  list  get global colors
                                                                index  1
    set VerticalMiddleLabel . BackgroundColor to  select list item  list  get global colors
                                                                  index  2
    set VerticalRightLabel . BackgroundColor to  select list item  list  get global colors
                                                                 index  3
```

  - We do so in every case the flag consists of vertically arranged stripes by retrieving the data of the appropriate country.
- Find out more about *TinyDB* by clicking on the links below.
  - http://www.appinventor.org/assets/pdf/ch22Databases.pdf
  - http://appinventor.pevest.com/?p=58

Task 3 | **MIT App Inventor 2** | ***"Where's my car"***

- The app demonstrates how to communicate with the Android location sensor, how to record data in the phone's long-term memory (database), and how you can open the Google Maps app from your app to show directions from one location to another. It makes use of the following App Inventor components:
    - *Location Sensor*
    - *TinyDB* //to store the data
    - *ActivityStarter* //to open a map
- Here are the components for the *Where's My Car?* app, as shown in the *Component Designer:*



Action: android.intent.action.VIEW
ExtraKey:
ExtraValue:
ResultName:
DataUri:
ActivityPackage: com.google.android.apps.m
ActivityClass: com.google.android.maps.M

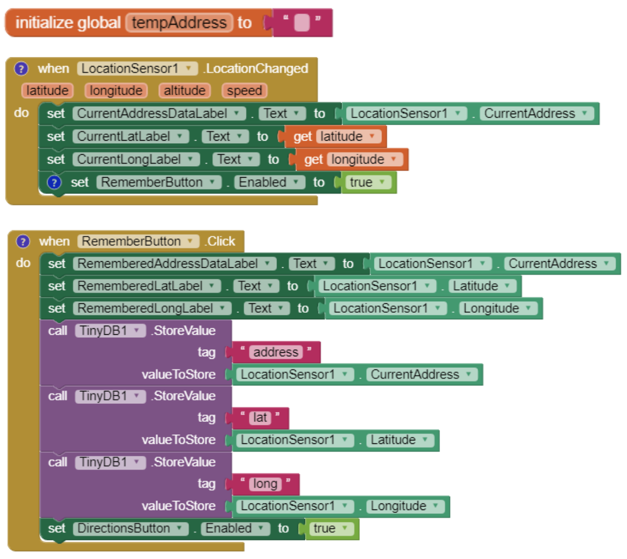
- The *ActivityStarter1* component is used to launch the map when the user asks for directions. Its properties are only partially shown above. Here is how they should be specified:

Property	**Value**
Action	android.intent.action.VIEW
ActivityClass	com.google.android.maps.MapsActivity
ActivityPackage	com.google.android.apps.maps

- Let's examine the four different event-handlers of the app, starting in the top-left and working around in counter-clockwise order.
    - LocationSensor1.LocationChanged: This event occurs when the phone's location sensor first gets a reading, or when the phone is moved to produce a new reading.

The event-handler just places the readings – latitude, longitude, and current (street) address – into the corresponding *'Current'* labels so that they appear on the phone. The *RememberButton* is also enabled in this event-handler. Its enabled setting should be unchecked in the *Component Designer* because there is nothing for the user to remember until the sensor gets a reading.

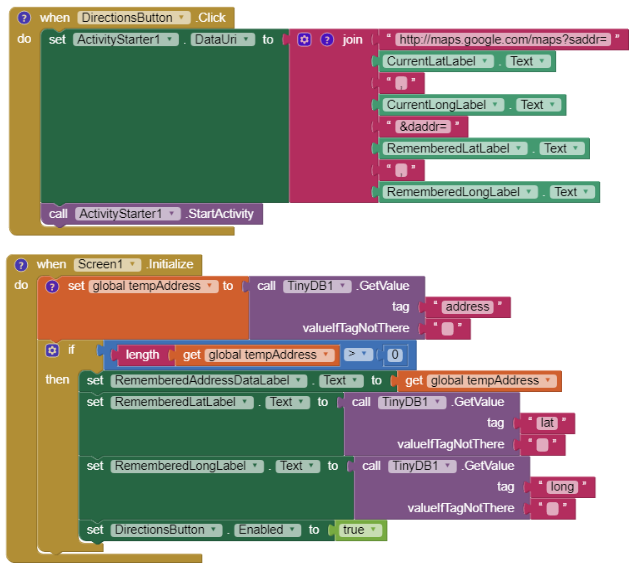
- RememberButton.Click: When the user clicks the *RememberButton*, the location sensor's current readings are put into the *'remember'* labels and stored to the database as well. The *DirectionsButton* is enabled as it now makes sense for the user click on it to see a map (though it will make more sense once the user changes location).
- DirectionsButton.Click: When the user clicks the *DirectionsButton*, the event-handler builds a URL for a map and calls *ActivityStarter* to launch the Maps application and load the map. join is used to build the URL to send to the *Google Maps* application. The resulting URL consists of the *Google Maps* domain along with two crucial parameters, *saddr* and *daddr*, which specify the start and destination for the directions. For this app, the *saddr* is set to the latitude and longitude of the current location, and the *daddr* is set to the latitude and longitude of the location that was *'remembered'*.
- Screen1.Initialize: This event is always triggered when an app opens. To understand it, you have to envision the user recording the location of the car, then closing the app, then later re-opening the app. When the app re-opens, the user expects that the location remembered earlier should appear on the phone. To facilitate this, the event-handler queries the database (call TinyDB.GetValue). If there is indeed a remembered address stored in the database – the length of the stored address is greater than zero – the remembered latitude, longitude, and street address are placed in the corresponding labels.



```
initialize global tempAddress to " "

when LocationSensor1.LocationChanged
  latitude longitude altitude speed
do set CurrentAddressDataLabel.Text to LocationSensor1.CurrentAddress
   set CurrentLatLabel.Text to get latitude
   set CurrentLongLabel.Text to get longitude
   set RememberButton.Enabled to true

when RememberButton.Click
do set RememberedAddressDataLabel.Text to LocationSensor1.CurrentAddress
   set RememberedLatLabel.Text to LocationSensor1.Latitude
   set RememberedLongLabel.Text to LocationSensor1.Longitude
   call TinyDB1.StoreValue
        tag "address"
        valueToStore LocationSensor1.CurrentAddress
   call TinyDB1.StoreValue
        tag "lat"
        valueToStore LocationSensor1.Latitude
   call TinyDB1.StoreValue
        tag "long"
        valueToStore LocationSensor1.Longitude
   set DirectionsButton.Enabled to true
```



```
when DirectionsButton.Click
do set ActivityStarter1.DataUri to join "http://maps.google.com/maps?saddr="
                                        CurrentLatLabel.Text
                                        ","
                                        CurrentLongLabel.Text
                                        "&daddr="
                                        RememberedLatLabel.Text
                                        ","
                                        RememberedLongLabel.Text
   call ActivityStarter1.StartActivity

when Screen1.Initialize
do set global tempAddress to call TinyDB1.GetValue
                                  tag "address"
                                  valueIfTagNotThere " "
   if length get global tempAddress > 0
   then set RememberedAddressDataLabel.Text to get global tempAddress
        set RememberedLatLabel.Text to call TinyDB1.GetValue
                                            tag "lat"
                                            valueIfTagNotThere " "
        set RememberedLongLabel.Text to call TinyDB1.GetValue
                                             tag "long"
                                             valueIfTagNotThere " "
        set DirectionsButton.Enabled to true
```

- Find out more about *Sensors* by clicking on the link below.
  http://www.appinventor.org/bookChapters/chapter23.pdf