USER MANUAL
GOOGLE EARTH ENGINE SCRIPT FOR SEMI-AUTOMATED MAPPING OF SMALL RESERVOIRS USING SENTINEL-2 SATELLITE IMAGERY

This user manual

Prepared by	Mary Amponsah
Date	December 2022
Supervision	Dr. Sander Zwart Dr. Komlavi Akpoti

Contents

Figures

1.0 BACKGROUND

This project uses Google Earth Engine since it has readily accessible, cost-free tools for the analysis to map small reservoir surface areas in time using Sentinel 2 imagery.

2.0 METHODOLOGY

The methodology used is summarized below. This includes the datasets used, flow chart and a step by step of the methodological steps.

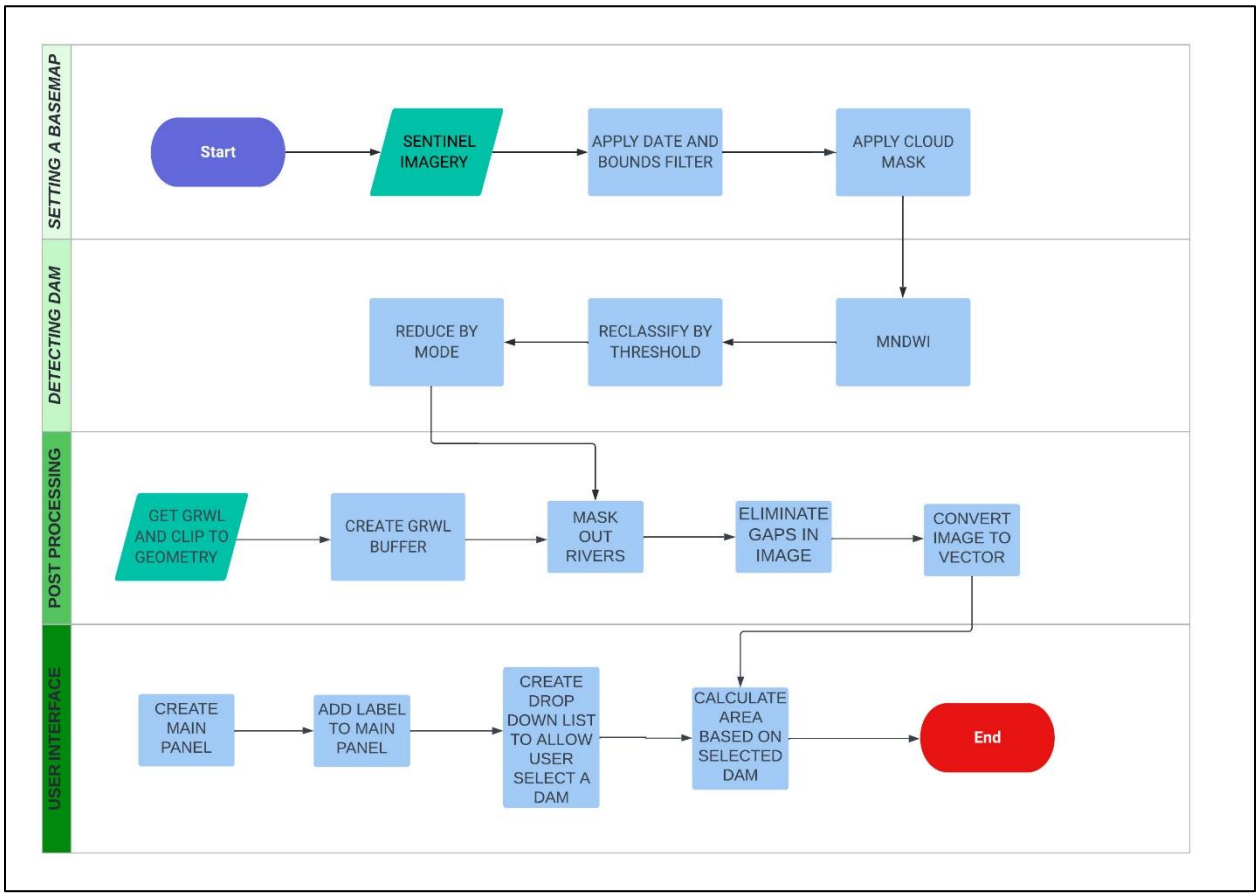

Figure 1. Flow chart for small reservoir mapping in Google Earth Engine Dam

2.1 SETTING A BASE MAP

- Filter the region boundaries by column name ‘REGION’ and row name ‘NORTH EAST’

```
var region = regbound.filter(ee.Filter.eq('REGION', 'NORTH EAST'));
```

- Set date range to be used to filter image collection.

```
var start_date = '2022-01-01';

var end_date = '2022-01-31';
```

- Define a function to filter out clouds using the 'SCL' band. Select the value 10 and 3 which represents clouds and cloud shadows respectively. Divide the image by 10000

```
function maskCloudAndShadowsSR(image) {

 var scl = image.select('SCL');

 var cloudBitMask = 1 << 10;

 var shadowBitMask = 1 << 3;

var mask = scl.bitwiseAnd(cloudBitMask).eq(0)

    .and(scl.bitwiseAnd(shadowBitMask).eq(0));

  return image.updateMask(mask).divide(10000);

}
```

- Call sentinel imagery filtered by dates and bounds. Apply the cloud mask function to the image

```
var S2_Image = ee.ImageCollection("COPERNICUS/S2_SR")

            .filterDate(start_date, end_date)

            .filterBounds(region)

            .map(maskCloudAndShadowsSR)
```

- Print the filtered sentinel imagery to check the number of images in its collection. If the number of elements in the collection is less than 2, widen the date range.

```
print(S2_Image)
```

- Define visualization parameters for the sentinel imagery

```
var rgbVis = {

 min: 0.066,

 max: 0.335,

 bands: ['B4', 'B3', 'B2'],

};
```

- Add the sentinel imagery to the map

```
Map.addLayer(S2_Image, rgbVis, 'S2 Image RGB');
```

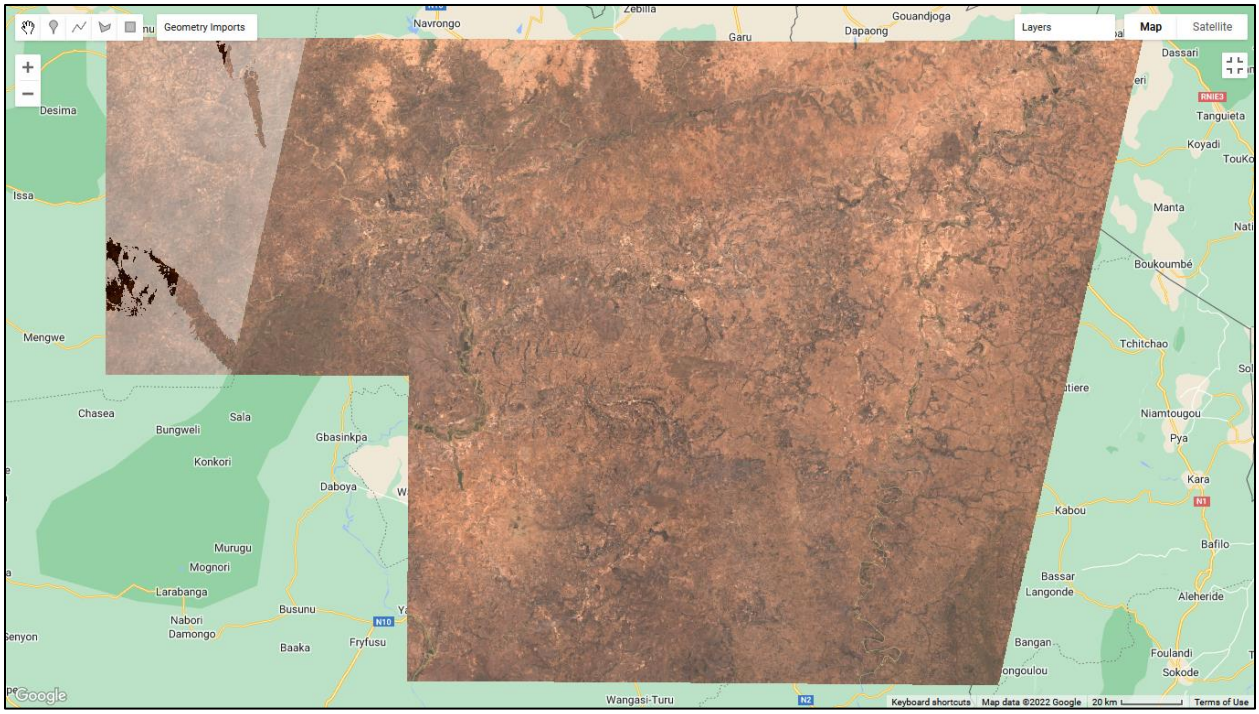

Figure 2. Sentinel 2 imagery over the North East region

2.2 DETECTING DAM

- Clip the sentinel imagery to the Dams of interest

```
var damBound_img = S2_Image.map(function(img){return img.clip(geometry)});
```

- Calculate and map Modified Normalized Difference Water Index (MNDWI) on the image and rename the band.

```
var mndwi = damBound_img.map(function(i){return i.normalizedDifference(['B3',
'B11']).rename('mndwi')});
```

- Check the MNDWI and add it the map.

```
print(mndwi)

Map.addLayer(mndwi, {}, 'MNDWI', false)
```

Figure 3. Sample of MNDWI over a selected dam

- Set out threshold to differentiate water from non-water area and reclassify the output.

```
var threshold = -0.13

var mndwi_classified = mndwi.map(function(i){return i

 .where(i.gte(threshold), 1)

 .where(i.lt(threshold), 0)

})
```

- Add the reclassified output to the map

```
Map.addLayer(mndwi_classified.map(function(i){return i.selfMask()}), {palette:
'#1bfff4'},'mndwi_classification', false)
```

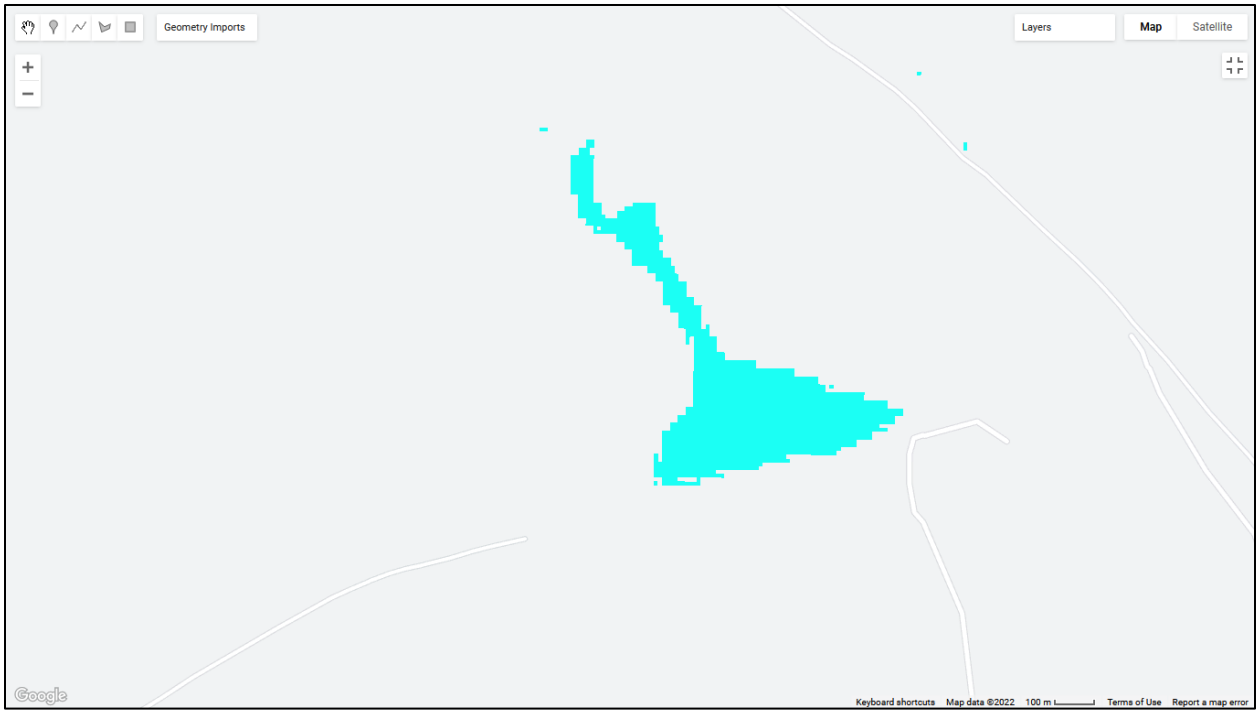

Figure 4. Reclassified image using a threshold value over the MNDWI image

- Reduce the reclassified output to get the most frequent value for each pixel. This is done to filter out flooded regions. Clip the output to the Dams and add it to the map.

```
var water_occurrance = mndwi_classified.mode();

var MaxWaterOccurance = water_occurrance.clip(Dams);

Map.addLayer(MaxWaterOccurance.selfMask(), {palette: '#1653ff'},'water_occurrance', false)
```

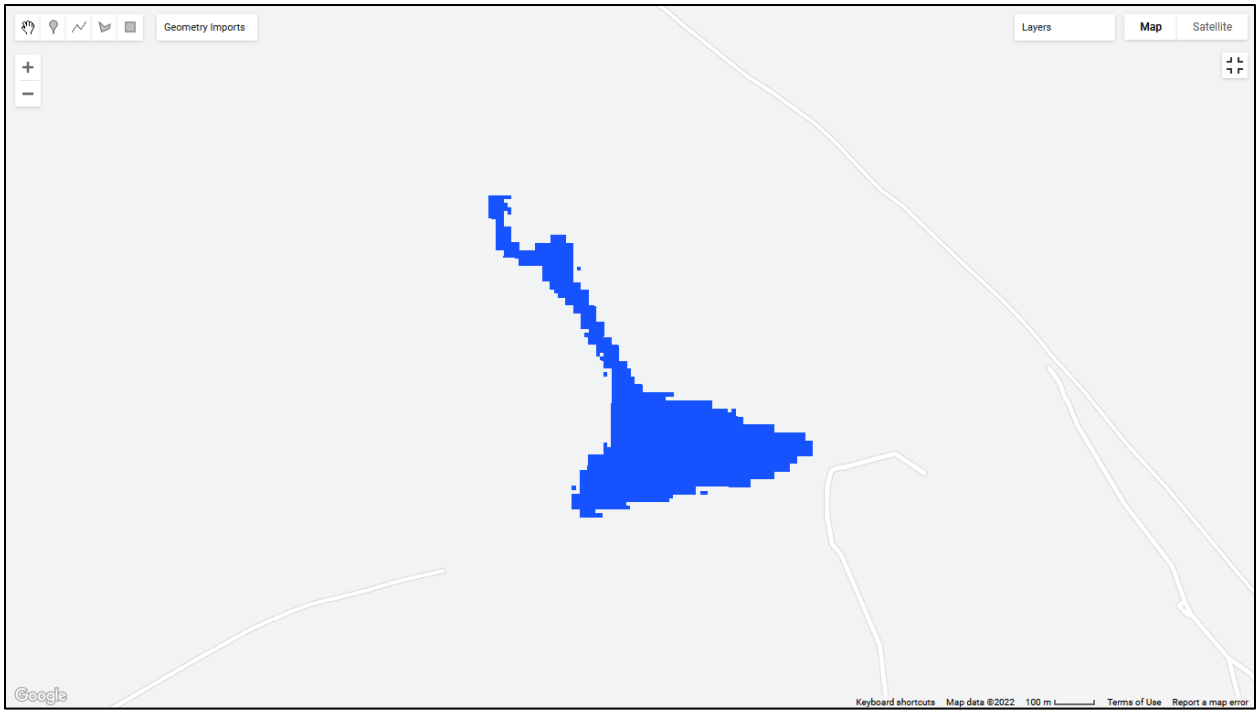

Figure 5. Maximum water occurrence throughout the specified month

2.3 POST PROCESSING OF DETECTED DAMS

- Call out Global River Width from Landsat (GRWL) and clip it to the geometry. Add GRWL dataset to the map

```
var global_watermask = ee.ImageCollection('projects/sat-io/open-datasets/GRWL/water_mask_v01_01');

function clp(img) {

 return img.clip(geometry);

}

var rivermask = global_watermask.map(clp).mosaic();

Map.addLayer(rivermask, {'min':11,'max':125,palette:['red']},'GRWL River Mask',false)
```

- Some rivers in the reduced image (MaxWaterOccurance) falls outside the bounds of the GRWL dataset. Create a buffer to extend the bounds of GRWL to include those rivers. Add the masked output to the map.

```
var riverbuffer = ee.Image(1)
  .cumulativeCost({
    source: rivermask,
    maxDistance: 100,
  }).lt(100);
Map.addLayer(riverbuffer.mask(riverbuffer), {min: 0, max: 100, palette: ['red']}, 'River_buffer', false);
```

- Mask out the buffered rivers from the reduced image. Apply the selfMask function to eliminate 0 value pixels. Print it out to the console and add it to the map.

```
var SmallReservoir = MaxWaterOccurance.updateMask(riverbuffer.unmask().not()).selfMask()
Map.addLayer(SmallReservoir, {min:0 , max:1, palette:['red']}, 'Reservoirs', false);
print(SmallReservoir,'SmallReservoir')
```

-
- Convert the previous output to a 16 bit integer.

```
var reservoir = SmallReservoir.uint16()
```

- Replace the masked pixels with the value 0

```
var reservoir = reservoir.unmask(0)
```

- A focal min and max is applied to fill the holes in the imagery caused as a result of masked out clouds.

```
var damProcessed = reservoir
 .focalMax({
   'radius':10,
   'units': 'meters',
   'kernelType': 'square'})
 .focalMin({
   'radius':10,
   'units': 'meters',
   'kernelType': 'square'});
```

- The filled image is added to the map.

```
Map.addLayer(damProcessed.selfMask(), {min:0, max:1, palette: ['blue']}, 'Surface Water',false)
```

- The filled image is converted to vector format and added to the map

```
var damVector = damProcessed.select(0).reduceToVectors({

 reducer: ee.Reducer.countEvery(),

 geometry: geometry,

 geometryType:'polygon',

 scale: 10,

 maxPixels: 1e10,

 eightConnected: false

})

Map.addLayer(damVector,{color: 'blue'}, 'Surface Water Polygons')
```

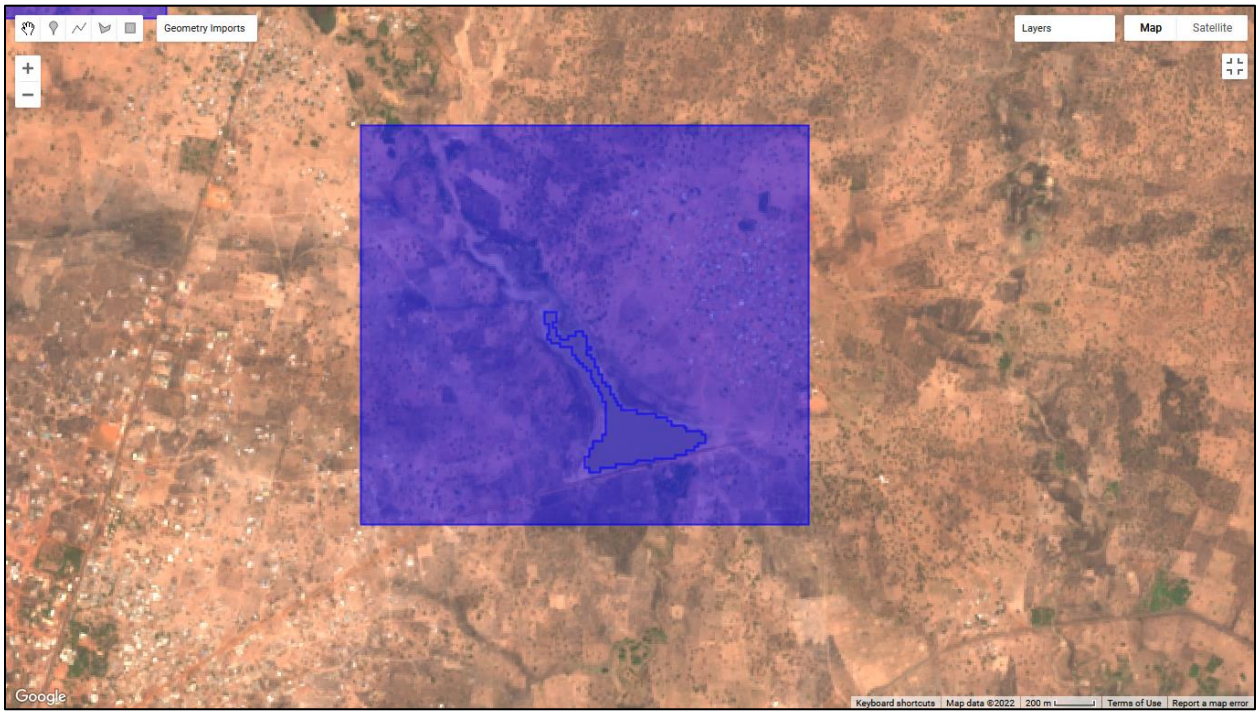

Figure 6. Output of post processed detected dam

2.4 CREATING A USER INTERFACE

- Create a main panel which will contain all widgets

```
var mainPanel = ui.Panel({

 style: {width: '300px'}

});
```

- Create a title for the main panel and add it to the panel

```
var title = ui.Label({

 value: 'DAMS AND THEIR AREAS',

 style: {'fontSize': '24px'}

});

mainPanel.add(title)
```

- Define a label to contain the results of the calculated area

```
var areaLabel = ui.Label();
```

- Create a list containing names of dams

```
var damNames = Dams.aggregate_array('Dam_Name');
```

- Create a drop down menu from which to select a dam name from. Define a function containing commands to execute whenever a dam is selected

```
var dropDown = ui.Select({

 placeholder: 'Select a Dam',

 items: damNames.getInfo(),

 onChange: function(value){
```

2.5 COMMANDS TO EXECUTE WHEN A DAM IS SELECTED

- Filter the geometry of the selected dam from the Dams shapefile

```
var selected = Dams

     .filter(ee.Filter.eq('Dam_Name', value))

     .first();

   print(value)
```

- Clip the selected dam to the ‘SmallReservoir’ output

```
var selectedDam = SmallReservoir.clip(selected);
```

- Calculate the area of each image pixel. Reduce the image by summing up the values of each pixel to get the area of the selected dam in meters. Print the calculated area.

```
var damArea = selectedDam
    .multiply(ee.Image.pixelArea())
    .reduceRegion({
     reducer: ee.Reducer.sum(),
     geometry: geometry,
     scale: 10,
     maxPixels: 1e9
     });
  print(damArea)
```

- The output of the calculated area is in a dictionary format. Change the dictionary to a number to convert the meters to hectares. Round up the values to the nearest whole number.

```
  var Area = ee.Number(damArea.get('mndwi')).divide(10000).round();
```

- Get the value of the area, add supplementary text to it and display it to the areaLabel.

```
  Area.evaluate(function(result) {
    var text = 'Dam Area : ' + result + ' Hectares'
    areaLabel.setValue(text)
  })
```

- Zoom in to the selected dam. Close all brackets.

```
  Map.centerObject(selected);
  }
});
```

- Create a supplementary text to the drop down list of the names of dams.

```
var damLabel = ui.Label({
 value: 'Select a Dam:',
 style: {'fontSize': '18px'}
});
```

- Visualize the supplementary text and the drop down list to be horizontal (next) to each other.

```
var damTitles = ui.Panel(
                [damLabel,
                dropDown],
               ui.Panel.Layout.Flow('horizontal'),
    {stretch: 'horizontal'})
```

2.6 FINALIZING USER INTERFACE

- Add the visualized supplementary text and drop down list to the main panel. Also add the areaLabel containing the area of the selected dam to the main panel.

```
mainPanel.add(damTitles)

mainPanel.add(areaLabel)
```

- Finally add the main panel to the map

```
ui.root.add(mainPanel);
```

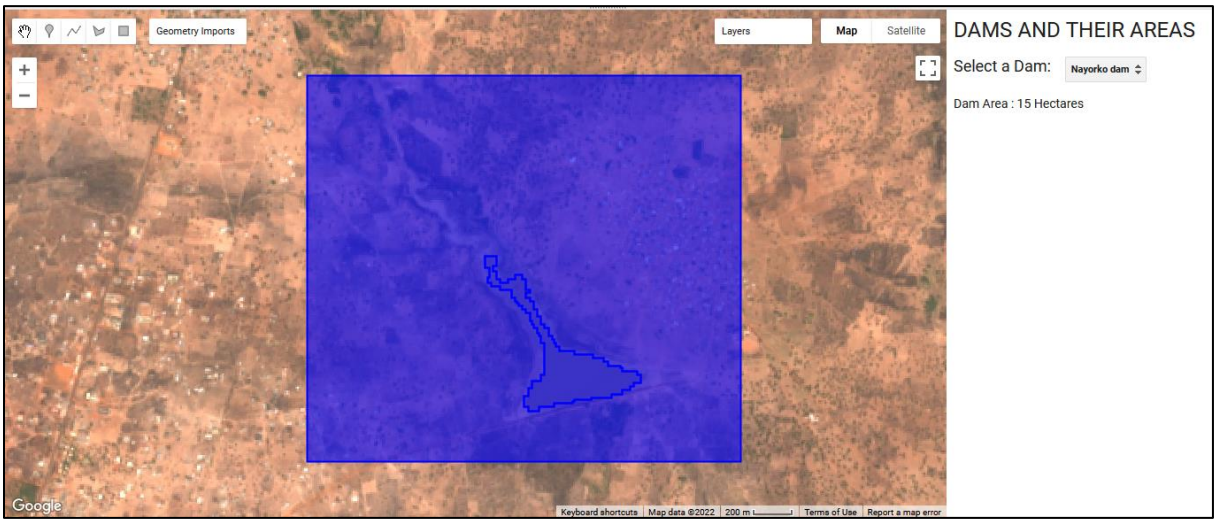

Figure 7. Finalized map with user interface together with the area of a selected dam