



Miikka Kallio

Interactive Mapping Tool For Renewable Energy Resources

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

6 May 2022

Tiivistelmä

Tekijä: Miikka Kallio
Otsikko: Interaktiivinen uusiutuvien energiaressurssien kartoitustyökalu
Sivumäärä: 33 sivua
Aika: 6.5.2022

Tutkinto: Insinööri (AMK)
Tutkinto-ohjelma: Tieto- ja viestintätekniikka
Ammatillinen pääaine: Mobile Solutions
Ohjaajat: Lehtori Ilkka Kylmäniemi
Tutkija Marko Kallio, Aalto-yliopisto
Pääjohtaja Tarek Ketelsen, AMPERES

Insinööriyössä toteutettiin selainpohjainen prototyyppisovellus, joka kartoittaa uusiutuvia energialähteitä Myanmarissa sijaitsevan eteläisen Shanin osavaltion alueella. Sovellus yhdistää kysyntäpisteet (asutukset) ja saatavilla olevat uusiutuvat energiaressurssit käyttäjän valitseman paikan mukaisesti.

Työssä dokumentoitiin sovelluksen käytössä olevat avoimet datasarjat, toteutettiin tiedonhaku merkityksellisistä maantieteellisistä piirteistä kahdella tavalla sekä huomioitiin jatkokehittämisen kannalta tärkeimmät ongelma-alueet.

Työn tuloksena saatiin toimiva prototyyppi, joka kartoittaa Shanin osavaltion alueella kuukausittaisen tuulen nopeuden ja aurinkovoiman potentiaalin sekä jokiverkostot. Sovellusta voidaan hyödyntää uusiutuvaan energiaan pohjautuvien piensähköverkkojen soveltuvuuden arvioinnissa.

Avainsanat: GIS, Leaflet, PostGIS, PostgreSQL, QGIS, kartta, interaktiivisuus, energia, CORE-KIT, Myanmar

Abstract

Author: Miikka Kallio
Title: Interactive mapping tool for renewable energy resources
Number of Pages: 33 pages
Date: 6 May 2022

Degree: Bachelor of Engineering
Degree Programme: Information Technology
Professional Major: Mobile Solutions
Supervisors: Ilkka Kylmäniemi, Principal Lecturer
Marko Kallio, Researcher, Aalto University
Tarek Ketelsen, Director General, AMPERES

This paper focuses on creating a prototype open-source web-application for mapping renewable energy resources in southern Shan State, Myanmar. The application connects demand points (settlements) with the available resources based on the location the user selects. The thesis documents the datasets used in the prototype, implements two methods for acquiring data from relevant features, and highlights problem areas for further development.

As a result of this study, a functioning prototype mapping tool was created. The prototype maps wind, solar and hydro resources within southern Shan State. The application can be used as a tool to conduct prefeasibility analysis for developing mini-grids for rural villages that currently do not have access to reliable electricity.

Keywords: GIS, Leaflet, PostGIS, PostgreSQL, QGIS, Map, Interactivity, Energy, CORE-KIT, Myanmar

Table of Contents

List of Abbreviations

1	Introduction.....	1
2	CORE-KIT, Geographic information system and database systems.....	3
2.1	What is CORE-KIT?.....	3
2.2	User case.....	3
2.3	Objectives.....	4
2.4	Geographic information system.....	4
2.5	Vector data.....	4
2.6	Raster data.....	5
2.7	GeoJSON and GeoTIFF.....	6
2.8	Database systems.....	7
2.9	PostgreSQL.....	8
2.10	PostGIS.....	9
3	Layer data.....	10
3.1	QGIS.....	10
3.2	Basemaps.....	10
3.3	Settlements.....	11
3.4	Districts and townships.....	12
3.5	Solar potential.....	13
3.6	Wind speed.....	14
3.7	Medium voltage grid.....	14
3.8	Rivers.....	14
3.9	Importing vector data to PostGIS.....	15
3.10	Importing raster data to PostGIS.....	16
4	Adding layers and interactivity to Leaflet.js map.....	17
4.1	Leaflet.....	17
4.2	React Leaflet.....	17
4.3	Creating Leaflet map and adding basemap layers.....	18

4.4 Adding GeoJSON overlays.....	18
4.5 Adding georaster overlays.....	20
4.6 Map interactivity.....	20
4.7 First Method: Bounds and iterating through vector features.....	22
4.8 Bounds and missing data problem.....	23
4.9 Second Method: SQL queries with spatial functions.....	25
5 Future development.....	28
5.1 User interface.....	28
5.2 Rasters.....	28
5.3 Estimating demand and electricity generation.....	28
5.4 Micro-hydro.....	30
5.5 Expanding coverage and uniform spatial reference system.....	31
6 Conclusion.....	32
References.....	34

List of Abbreviations

CORE-KIT The name of the prototype application, abbreviated from
Community Renewable Energy Knowledge Integration Tool

GIS A geographic information system.

SQL Programming language, abbreviated from Structured Query
Language.

SRID Unique identifier for specific coordinate systems, abbreviated from
Spatial reference identifier.

1 Introduction

Access to electricity has a direct impact on quality of life. The electrification of rural communities can improve health, education, security, and productivity by allowing access to such basics as lighting, refrigeration, recharging phones, and enabling water pumps among many other uses. By using renewable energy resources these benefits are accessible without negatively impacting the local ecosystems. [1.]

Myanmar has the poorest electrification rate in South East Asia, leaving approximately half of its population without access to reliable electricity. In rural areas of the country, 70 percent of the population relies on candles, kerosene, batteries, and power generators on a daily basis. Myanmar has significant potential resources for solar, wind, and hydropower. Across the country, numerous mini-grids are already in operation and further mini-grids could be used as an option for electrifying rural villages not connected to the national power grid.

The Ministry of Electricity and Energy (MOEE) has set a target of 76 percent electrification by 2025 and 100 percent electrification by 2030 under the National Electrification Project (NEP). [2.] Table 1 shows the status of NEP in Shan State, as of December 2021. This thesis focuses on the southern region of Shan State.

Table 1. Excerpt from the Myanmar National Electrification Project (NEP) showing the current electrification status of villages and households in Shan State, as of December 2021. [3.]

Description	Shan (South)	Shan (North)	Shan (East)
No. of existing villages	4784	5241	3748
No. of electrified villages (National grid)	1717	678	109
No. of electrified villages (Off-grid)	477	1253	768
Total no. of electrified villages	2194	1931	877
Remaining villages to electrify	2590	3310	2871
Total no. of households	514639	445983	208947
No. of households being electrified	333959	194351	42904
Percentage of households being electrified	64.89	43.58	20.53

In the four years between 2016 and 2020, 8568 villages were electrified using mini-grids and solar home systems, providing access to electricity for over 2 million people in Myanmar. [2.]

2 CORE-KIT, Geographic information system and database systems

2.1 What is CORE-KIT?

Community Renewable Energy Knowledge Integration Tool (CORE-KIT) is an open-source interactive web application for mapping renewable energy resources. The tool is being developed by The Australia Mekong Partnership for Environmental Resources and Energy Systems (AMPERES). Its goal is to provide a technical case for preliminary feasibility assessment for localized small-scale energy grids. It aims to cover relevant renewable energy resources including wind, solar, biomass, and micro-hydropower. At the time of writing, CORE-KIT is in its prototype development phase and covers wind and solar. Datasets used by the prototype are primarily focused on southern Shan State, Myanmar.

CORE-KIT connects demand points (settlements) with renewable energy resources, allowing the possibility to calculate and visualize rough estimates for both demand and energy production, based on the location selected. [4.]

2.2 User case

While CORE-KIT could potentially have a wide range of users and use cases, the most common expected user would be a member of a non-governmental organization (NGO) to conduct prefeasibility analysis of potential locations for developing mini-grids.

CORE-KIT presents the user with an interactive map, that allows the user to select a location on the map. The map includes various overlays displaying relevant data, such as settlements. Each overlay can be turned on or off through overlay controls on the map.

After selecting the location on the map, a marker is placed on the spot and the user is presented with general village and river flow data from the nearest features. In addition, the user is shown wind speed at 50 meters and solar potential kilowatt-hour per kilowatt peak (kWh/kWp) at the selected location. After studying the data the user would have a better understanding of the potential of the location and whether to invest in real-world data gathering. [4.]

2.3 Objectives

The goal of this study was to create an interactive web application to map demand points and wind, solar and hydrology resources in southern Shan State, Myanmar. The study consists of three main objectives. The first objective was to document the datasets used in the CORE-KIT web application and to record how the datasets were modified and imported into a database. The second objective was gathering data based on the selected location and presenting said data. The third objective was to identify key features that need to be developed or improved.

2.4 Geographic information system

Geographic information system (GIS) is a system that incorporates spatial information and descriptive attributes together. GIS is a constantly evolving technological field used by a large number of industries in some capacity. It can be used for mapping, performing spatial analysis, and assessing a wide range of issues around the world. Spatial data handled by GIS can be sorted into two categories: vector and raster data. [5.]

2.5 Vector data

Real-world features can be referred to in a GIS environment by utilizing vector data. Each vector feature is formed by a shape and descriptive attributes. Vector data is typically discrete, meaning that they can exist individually and

their boundaries are defined. Vector features can be sorted into three categories based on the dimensions of the feature forming its shape: point (zero-dimensional), line (one-dimensional), or polygon (two-dimensional).

The zero-dimensional feature is represented as a point and is formed by a single X, Y coordinate. Optionally the geometry may include Z and M coordinates. These coordinate values in most cases refer to latitude as X, longitude as Y, and altitude as Z, pointing to the location of the feature. [6.] The M coordinate (measure) can be included to store additional non-coordinate information within the geometry, such as distance or uncertainty. [7.] As an example, a point feature can be used for mapping points of interest and its attributes could include information such as type and name.

A one-dimensional feature, which is represented as a line is formed by drawing a continuous line through two or more positions. This type of geometry is often also referred as polyline or linestring. These are used for linear features. As an example, a line feature could be used to depict a segment of a river and its attributes could contain discharge information.

Two-dimensional feature or a polygon is an enclosed area consisting of a collection of positions connected by a continuous line forming a ring, where the first and last positions are the same. [6.] A polygon can contain multiple outer and inner rings. Polygon features could depict administrative boundaries or country boundaries and its attributes could contain name and population.

2.6 Raster data

Raster data (also called grid data) is a matrix of cells (also called pixels), in which each cell contains information about the area covered by that cell. In GIS the raster is likely to be georeferenced or geocoded and thus each cell represents an area or surface. [8.] Rasters can be categorized based on the type of dataset to the following: thematic data, spectral data, and pictures (imagery). [9.]

Rasters are used for conveying information over a continuous area. While the vector polygon feature also refers to an area it is only able to store a single value over the whole area. The same area as a raster could contain many cells, thus greatly increasing the accuracy of the data. The size of each cell is fixed and defines the resolution of the raster.

Raster includes one or more layers of bands. The number of bands specifies the spectral resolution of the raster. Typically images contain three bands: red, green, and blue (RGB). However, the image may also include data for the non-visible light spectrum. This means that each cell contains one or multiple values based on the spectral resolution.

A single band raster is referred to as a grayscale image, whereas a single band raster with false coloring is referred to as a pseudocolor image. Multi-spectrum image refers to a raster image that includes the non-visible spectrum. [8.]

2.7 GeoJSON and GeoTIFF

GeoJSON is a geospatial data format based on JavaScript Object Notation (JSON) and is used for storing collections of geographical features by combining multiple types of JSON objects. The spatial data of a Feature is defined in a geometry object. GeoJSON supports Simple Features Standard, that covers all the vector geometries (point, polyline, and polygon) and their multi-part variants. The multi-part variants are as follows:

- MultiPoint refers to a feature consisting of multiple Points.
- MultiLineString refers to a feature consisting of multiple LineStrings (polylines).
- MultiPolygon is a feature that is formed by multiple Polygons.

GeoJSON also supports GeometryCollections, which are collections of various geometric types. The shapes are described by single or sets of coordinates consisting of latitude, longitude, and optionally elevation.

In addition to its shape, a Feature contains defining non-spatial properties. These are stored in a properties object, which consists of combinations of keys and their values. [10.] Listing 1 shows an example of how a single point type GeoJSON Feature describing a settlement could be formed.

```
{ "type": "Feature",
  "properties": {
    "fid": 1,
    "Village": "singtaung",
    "Village_HH": "170",
    "Population": "1090"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      97.095422,
      19.7099567
    ]
  }
}
```

Listing 1. Example of a GeoJSON Feature formed by combining settlement data as properties, and point type geometry containing its geographical location.

In addition, typically a GeoJSON file includes a FeatureCollection. FeatureCollection is an object that holds an array of individual feature objects. [11, p. 12]

GeoTIFF is a Tagged Image File Format (TIFF) developed by Niles Ritter. It differs from a regular TIFF file by containing georeferencing information within the file, whereas TIFF requires an accompanying .tfw file when used in a GIS environment. [12.]

2.8 Database systems

Databases are collections of logically coherent data that represent aspects of the real world. They are built and populated with data for a specific purpose. Database management system (DBMS) is software that allows users to create and manage databases. DBMS enables applications to access the database by queries. These queries can request specific data, perform an action on the data

or do both. DBMS also provides protection against hardware and software malfunctions or unauthorized access. [13, p. 4-6.]

Relational database management system (RDBMS) utilizes the relational data model. In this model, each table in the database is considered a relation. Each relation consists of attributes and tuples. Each column in the table is considered an attribute, and each row in the table is considered a tuple. [13, p. 150-151.]

Object database management system (ODBMS) utilizes object data model. In this model, the data is stored as objects. Each object contains attributes that represent its characteristics and its behavior is defined by methods. Objects are grouped together into a class based on their attributes and methods. Thus each object is considered an instance of the class they belong to. When a new class with unique characteristics or behavior is created, it would gain attributes and methods from the class it was derived from. This feature is referred to as inheritance [14.]. For example, a city class in addition to its own attributes would inherit the attributes from a settlement class.

Object-relational database management system (ORDBMS) incorporate capabilities from both data models. ORDBMS surfaced as developers of RDBMS incorporated features of ODBMS to close the gap between RDBMS and practices used frequently in object-oriented programming languages such as C++ and Java. [13, p. 363-364]

2.9 PostgreSQL

PostgreSQL is an open source ORDBMS with over 30 years of active development. The roots of PostgreSQL date back to the POSTGRES project at the University of California at Berkeley, led by Professor Michael Stonebraker in 1986. At the time of writing, the latest version of PostgreSQL was 14.1.

PostgreSQL utilizes and extends the Structured Query Language (SQL), and runs on all major operation systems. It is highly extensible, accommodating features such as:

- Stored functions and procedures
- Procedural languages
- SQL/JSON path expressions
- Foreign data wrappers
- Extensions

Stored functions and procedures allows the user to define their own functions using procedural languages (PLs). PostgreSQL supports many such languages, including PL/pgSQL, Perl, Python and many more. SQL/JSON path expressions allow querying JSON data stored in the database. Foreign data wrappers can be used to access data in external PostgreSQL servers. PostgreSQL allows extensions to be loaded into the database. These extensions provide additional functionality. [15.]

2.10 PostGIS

PostGIS is an open source spatial database extension for PostgreSQL developed by Refractions Research. It was first released for the public in 2001. It is OpenGeospatialConsortium's (OGC) Simple Features for SQL Specification compliant.

PostGIS adds the support for spatial types, spatial indexes, and spatial functions. This enables the ability to combine the SQL with spatial functions and operators. For example allowing filtering and selecting wanted objects by coordinates or attributes using standard SQL queries. PostGIS is well supported by both open source and proprietary third-party programs. The combination of PostgreSQL and PostGIS makes a robust and efficient open source spatial database for mapping applications and geographic information systems [16, p. 3].

3 Layer data

3.1 QGIS

Quantum GIS (QGIS) is a free open source desktop geographic information system. It allows the user to view, edit and analyze geospatial data. QGIS supports raster, vector, mesh and point cloud data in many industry-standard formats. It can access data from multiple sources including local files, spatial databases and web services. QGIS is extensible with a variety of plugins that add or extend features.

QGIS runs on Linux, Unix, Mac, Windows and Android. It is an official project of the Open Source Geospatial Foundation (OSGeo) and is maintained by a team of volunteers and organisations. [17.]

QGIS was used to modify the datasets used in CORE-KIT, by reducing the datasets to only include features located in southern Shan State. QGIS was also utilized to convert the original files from GeoJSON to ESRI Shapefile when needed to simplify the importing of the files to the database.

3.2 Basemaps

Basemaps are reference maps that serve as the base on which overlay data is added onto. For the CORE-KIT project, four different basemaps were selected, each providing slightly differing context. The base tilesets in use are: OpenStreetMap Standard, Mapbox Streets, Mapbox Satellite and Mapbox Satellite-Streets.

OpenStreetMap Standard serves as an exemplar general purpose basemap tileset. It's design goals are clarity, diversity, richness, maintainability and ease of use. [18.] Mapbox Streets is very similar to OpenStreetMap Standard, but provides additional visual topological information. Mapbox Satellite is a satellite imagery tileset, provided by National Aeronautics and Space Administration

(NASA) and United States Geological Survey (USGS). Mapbox Satellite-Streets is a combination of the satellite imagery and vector features.

3.3 Settlements

The settlement data used in CORE-KIT is the Myanmar – Populated Settlements Data, collected by Sustainable Engineering Lab and Columbia University Earth Institute from multiple sources such as: Ministry of Agriculture, Livestock and Irrigation Department of Rural Development (DRD), General Administration Department (GAD), and Myanmar Information Management Unit (MIMU). The data is available in three formats, Comma Separated Values (CSV), GeoJSON and Esri Shapefile. [19.] The dataset was published in 2014 and is available at Energydata.info website (<https://energydata.info/dataset/myanmar-populated-settlements-data>).

Each settlement contains properties including state, name, number of households and population. Figure 1 shows how selecting wanted features was achieved by using the Select Features by Expression function in QGIS.

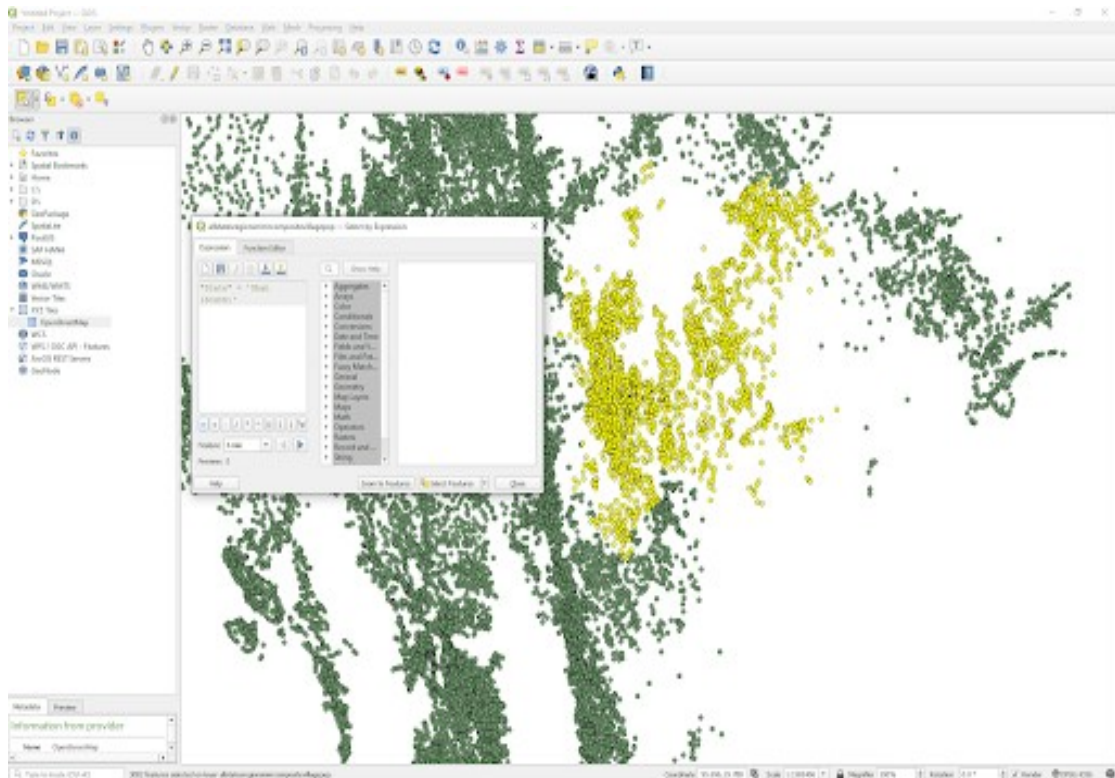


Figure 1. Select Features by Expression where the yellow highlighted features are selected based on the property "State" having the value of "Shan (South)".

After the wanted settlements had been selected they were copied into a new layer, saved and exported as a separate Esri Shapefile to be imported into PostgreSQL database.

3.4 Districts and townships

The district and township boundaries data in use are from Myanmar Information Management Unit (MIMU). The datasets are Myanmar Township Boundaries MIMU v9.3 and Myanmar District Boundaries MIMU v9.3. Both of them are downloadable as a layer or as metadata, in multiple formats. The data consists of vector polygon features, each feature containing properties including geometry, and name in both Myanmar Unicode and roman script. The datasets are available at the MIMU website (<http://geonode.themimu.info/layers/?limit=100&offset=0>).

3.5 Solar potential

The solar potential raster used in CORE-KIT is a thematic raster that combines twelve separate daily average photovoltaic electricity potential rasters. Meaning that each cell of the raster in use includes twelve values, one for each month of the year. The raster in use also only covers the southern Shan State, Myanmar, unlike the source rasters that cover the entirety of Myanmar. The raster used in CORE-KIT can be seen in Figure 2 inspected using QGIS.

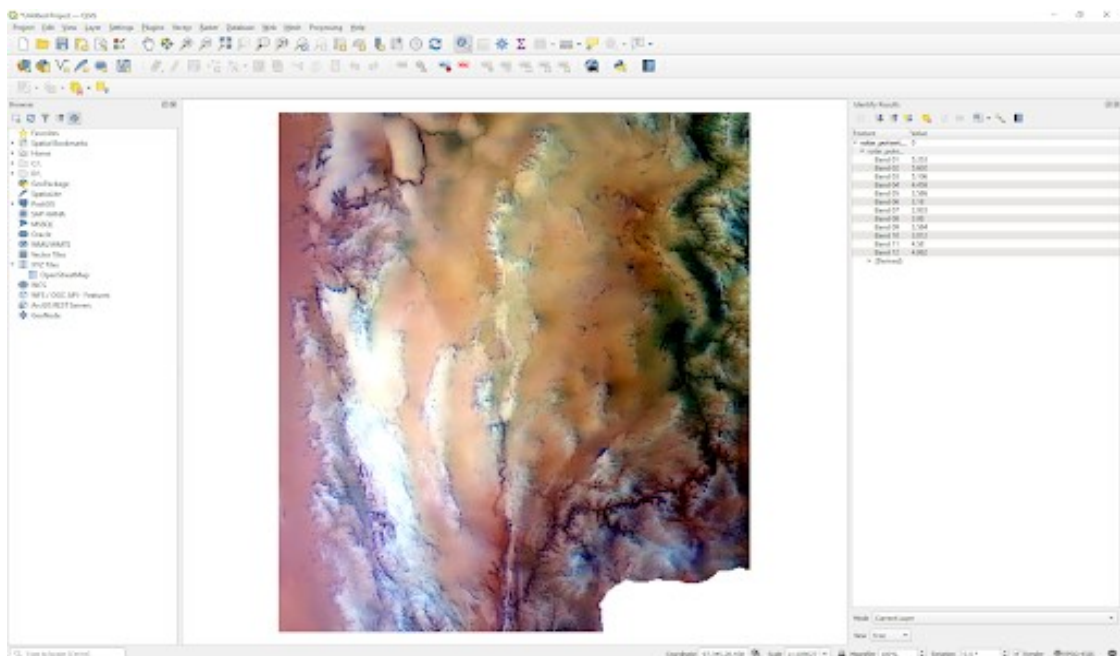


Figure 2. Solar potential GeoTIFF consists of cells containing daily average photovoltaic electricity potential (kilowatt hours per kilowatt peak, kWh/kWp) for each month as bands.

The source rasters are available for free from Solargis website (<http://solargis.com/maps-and-gis-data/download/myanmar>) under the Creative Commons Attribution license.

3.6 Wind speed

The wind speed data used in CORE-KIT was created by Andrew Said. The data is in raster form in which cells contain the average wind speed of each month of the year. The datasets used to create the raster are the following:

- Historic wind measurements between 1998 and 2018 from 32 weather stations.
- World Climate Data historic wind measurements between 1960 and 1990 and between 1920 and 2018.
- Global Wind Atlas (GWA) annual average wind speed at 50 meters.

The monthly average wind speeds in the raster are averages of mean monthly wind speeds based on the historic wind measurements and the GWA dataset. [20, p. 30 – 35.] This data was then reduced by cropping to only cover southern Shan State using QGIS.

3.7 Medium voltage grid

CORE-KIT uses existing medium voltage grid data collected by Earth Institute and Sustainable Engineering Lab in collaboration with the Resource and Environment Myanmar (REM). The dataset consists of 11 kV, 33 kV and 66 kV medium voltage lines. The dataset was published in 2014 and is available through Energydata.info website (<https://energydata.info/dataset/myanmar-existing-grid-medium-voltage-line-data>).

3.8 Rivers

The river dataset used in CORE-KIT is a cropped version of HydroRIVERS Version 1.0 for Asia. The dataset includes global rivers with catchment area of at least 10 km² or an average river flow of 0.1 cubic meters per second. The original dataset is available through HydroSHEDS website (<https://www.hydrosheds.org/page/hydrorivers>).

3.9 Importing vector data to PostGIS

Each of the vector datasets were added to PostgreSQL as their own table using PostGIS's Shapefile Import / Export Manager. If the source data needed to be reduced or if it was not already in shapefile format it was first converted into such using QGIS. Figure 3 shows importing of the township data.

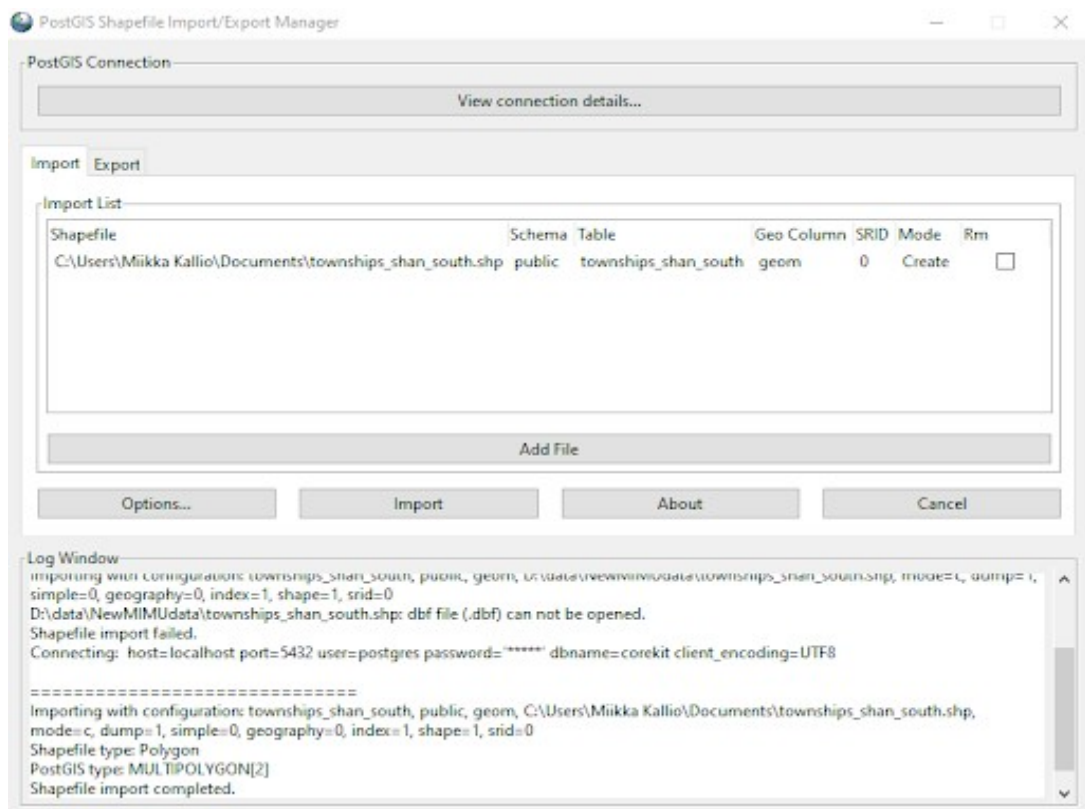
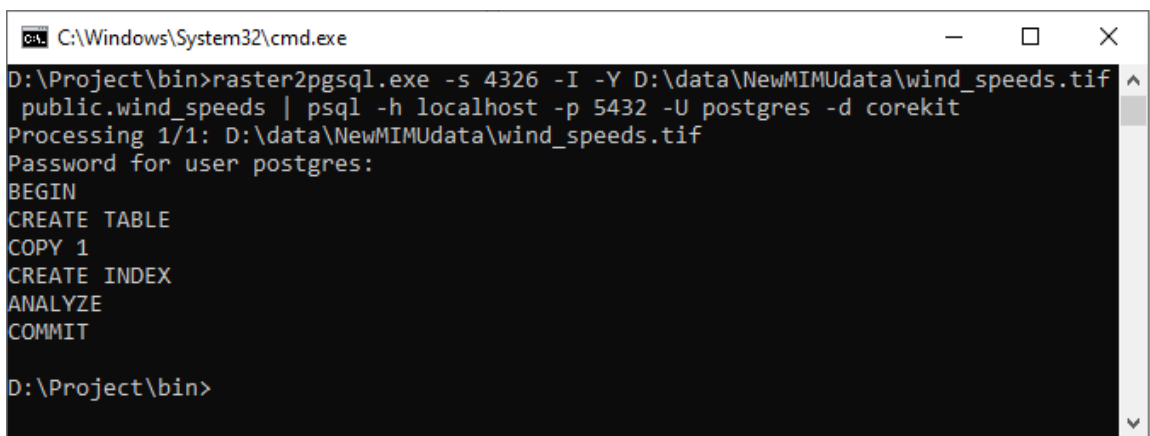


Figure 3. Importing townships_shan_south shapefile to PostGIS enabled PostgreSQL database.

In addition to the source shapefile, a database connection was required. The connection was established after defining the host, port number, database user, its password and the target database. This process was repeated with each vector dataset.

3.10 Importing raster data to PostGIS

To import raster data to the database, raster2pgsql was used. Raster2pgsql is an executable raster loader usable for importing Geospatial Data Abstraction Library (GDAL) supported rasters into PostgreSQL. It is included as a part of PostGIS. [21.] Figure 4 shows raster2pgsql utilized to import wind_speeds raster into PostgreSQL database.



```
C:\Windows\System32\cmd.exe
D:\Project\bin>raster2pgsql.exe -s 4326 -I -Y D:\data\NewMIMUdata\wind_speeds.tif
public.wind_speeds | psql -h localhost -p 5432 -U postgres -d corekit
Processing 1/1: D:\data\NewMIMUdata\wind_speeds.tif
Password for user postgres:
BEGIN
CREATE TABLE
COPY 1
CREATE INDEX
ANALYZE
COMMIT

D:\Project\bin>
```

Figure 4. Raster2pgsql command to import wind_speeds raster into corekit database.

The command seen in Image 4 runs a raster2pgsql executable with parameters. The -s parameter sets the spatial reference identifier (SRID), which should match the SRID of the source raster. SRID 4326 is WGS84 standard, used in Global Positioning System (GPS). The -I option creates a Generalized Search Tree (GiST) index on the raster column. The -Y parameter inserts the raster using copy statements instead of insert.

4 Adding layers and interactivity to Leaflet.js map

4.1 Leaflet

Leaflet is an open-source JavaScript map library. It's focus is on simplicity, usability and performance. Leaflet works well on all major mobile and desktop platforms. It is lightweight with a robust set of features. Leaflet is also extensible with a wide range of plugins. The library was originally created by Vladimir Agafonkin, but is now developed by a large community of contributors.

Leaflet can load a variety of basemap tile services and add vector feature layers atop those tiles using either Scalable Vector Graphics (SVG) or Canvas Application Programming Interface (API). Leaflet provides a variety of interaction features including drag panning, zoom and events such as mouseover and on click. [22.]

4.2 React Leaflet

React Leaflet allows the creation of Leaflet layers as React components. It does not replace Leaflet, causing React Leaflet components to differ in behavior compared to other React components in the following aspects:

- Document object model (DOM) rendering
- Properties
- Context

Layers are not rendered to the DOM by React, but rather by Leaflet. The layer's properties are considered immutable by default and will not update on change, unless documented as mutable. React-Leaflet uses React's context API for making instances of Leaflet elements available for child elements. [23.]

4.3 Creating Leaflet map and adding basemap layers

Creating a Leaflet map using React-Leaflet is accomplished by creating a `MapContainer` component. `MapContainer` is given properties, such as default zoom level, coordinates to center on, whether to prefer Canvas over SVG to render vector layers, and most importantly its fixed height. The `MapContainer` also holds relevant child components that are rendered on the map, notably `LayerControls`, `Popup` and `Legend`.

The basemap layers are added into `LayersControl.BaseLayer` in a separate `Layers` component. These basemap tiles are acquired by loading them from tile providers such as Mapbox and OpenStreetMap (OSM). OSM tiles are free, while Mapbox tiles are free for a set number of requests per month.

4.4 Adding GeoJSON overlays

Each overlay layer is created inside its own React component. Each of them contain functions such as `getData` and `render`. Vector layers also contain `onEachFeature` function and in addition, point type layers contain `pointToLayer` function. Using these functions a GeoJSON layer component can be created and added to the `LayersControl`, that the user can use to toggle overlays on and off.

To add an overlay to the Leaflet's `OverlayControl`, the data needs to be retrieved from the database. The data is acquired by sending a request to the server. The data is then received from the database as a response to an SQL query. Listing 2 shows the SQL query to receive settlement data.


```

SELECT jsonb_build_object(
    'type', 'FeatureCollection',
    'features', jsonb_agg(feature)
)
FROM (
    SELECT jsonb_build_object(
        'type', 'Feature',
        'gid', gid,
        'geometry', ST_AsGeoJSON(geom)::jsonb,
        'properties', to_jsonb(inputs) - 'gid' - 'geom')
    AS feature
    FROM (
        SELECT * FROM public.settlements
    ) inputs
) features;

```

Listing 2. SQL query to retrieve the data from settlements table from public schema in a GeoJSON format.

Once the data has been received a reference for the layer is created. The data and the reference are set as state variables so that they can be preserved outside the function when creating the GeoJSON layer.

The `onEachFeature` is a customizable function that each feature in the layer has access to. This is used on the polygon features, to push them to the bottom of the overlay stack when the layer is added to the map. This is achieved by calling `bringToBack()` function, as if the polygon layer is at the top of the overlay stack, it blocks interacting with underlying features. Polyline and point layers use the `onEachFeature` to enable mouse over interaction.

The `pointToLayer` function is called internally when the layer data is added. The function is used to define how the features spawn. In CORE-KIT each settlement point is assigned a color based on its population in four tiers and is returned as a circle instead of the default Marker. The four tiers of population are as follows: 0 to 99, 100 to 999, 1000 to 9999 and 10000 and higher. This is done so that the user has an idea of the size of the settlement without having to interact with it.

In the render function a style is set for the features contained in the layer. The style can be used to determine the color of the features. This is done with all layers with the exception of layers containing point type features. The render

function then returns the GeoJSON layer that includes the data, reference and any additional functions such as `onEachFeature`. The GeoJSON layer is then added into the overlay control.

4.5 Adding georaster overlays

As Leaflet does not support GeoTIFF format, a plugin such as `georaster-layer-for-leaflet` is required. The plugin provides a way to add GeoTIFF data as a layer onto the map. `Georaster-layer-for-leaflet` supports most projections, provides fast rendering and allows for custom cell coloring. [24.]

The georaster layers are created inside their own functional React components. At the time of writing the raster data is retrieved from an external Github repository. Once the raster has been received, the minimum value and range of each band is stored as a variable respectively.

To calculate the mean wind speed over the year, the wind speed value from each month of the year is summed then the sum is divided by twelve. The result of the equation is then stored as a `pixelValue` variable. If the value is negative or zero, no color is applied to the cell.

The minimum value is then deducted from the `pixelValue`, and the result is divided by the range value. The result of this equation ranges between zero and one. The value is then converted into a hexadecimal color using viridis palette. The calculations and color scaling is repeated for each cell in the raster.

4.6 Map interactivity

In addition to basic interactivity (panning, zooming, and layer controls) provided by Leaflet, additional interactivity was implemented. Point, and line features active on the map can be interacted by hovering over them. This causes them to display a popup containing the identifier of the feature in case of river segment, its population and name in case of a settlement feature, and in case

of medium voltage grid feature, its identifier and where the grid originates and its destination when available.

CORE-KIT also allows the user to select a location on the map by clicking the wanted location on the map. This action fires a custom function that opens a popup on the coordinates the user clicked on. The popup contains information about the relevant features. In addition, yearly average wind speed and solar potential at the location are contained in the popup.

The settlement, hydro, wind, and solar data are also presented in more detail on information tabs below the map. Each of them have their own tabs. The hydro tab at the time of writing only displays the identifier of the river segment.

In the settlement tab general information about the nearest settlement is displayed. The general information includes the name of the settlement, it's population and number of households.

Wind tab displays the average monthly wind speed at 50 meters, as a line graph created with Plotly.js, which is an open-source graphing library. The X-axis of the graph contains each month of the year and is titled as "Month". Y-axis contains the monthly wind data acquired when the map was clicked. The range of the ticks of the Y-axis are automatically scaled based on the wind speed values, and the axis is labeled as "Wind speed (m/s)".

Solar tab displays the daily- and monthly solar potential as line graphs. These graphs are nearly identical, only the data presented is changed. The X-axis of these graphs are identical to the wind speed graph as described above. Ticks on the Y-axis are scaled as in wind speed graph, however the title is changed to "kWh/kWp" (kilowatt hour per kilowatt peak). The data on the daily solar potential is the raw solar potential data acquired when the map was clicked. The data used on the monthly solar potential graph is modified from the data used on the daily graph by multiplying each value by 30, as CORE-KIT assumes each month having 30 days for simplicity.

To populate both the popup and information tabs with data from features, two methods were implemented. The first method used the active map layers and gathering data from the features overlapping with the coordinates the user clicked on by comparing bounds. Later on polygon features were acquired through querying the database. The second method used only database queries to find relevant features. Both of these methods utilized React state variables to store and pass the relevant information to the child components. The data tied to these variables are reset each time the map click interaction event is triggered. Each of the methods used outperformed the other in at least one aspect.

Both methods use SQL database queries to gather the wind speed and solar potential data at the selected location. The query selects the values of each band (month) contained in the cell, referring to the location where the map interaction occurred. The values are returned as a JSON object. The longitude and latitude values of the interaction's location are passed to the query through parameters. The data is then stored in state variables, one containing the values for each month of the year and another containing a single mean value.

4.7 First Method: Bounds and iterating through vector features

When the user clicks on the map, rectangular click bounds are created by using `latLngBounds`. The bounds are created by passing the latitude and the longitude as diagonally opposite corners of the rectangle. The longitude and latitude are taken from the location the click event occurred on. By passing the same coordinates to both of these corners, the bounding area is minimized. The click bounds are later compared to the bounds of each vector feature active on the map to find all overlapping features.

To iterate through each active feature on the map, a for loop was created. The loop went through each active layer on the map, checking if the layer contained features. Once a layer with features was found, a second for loop was created. The second loop iterated through each feature, getting their bounds and

checking if they overlap with the click bounds created earlier. When an overlapping feature was found the object was pushed into an array containing all overlapping features.

Once all overlapping features had been found and added to the array, they were tested against predetermined unique properties to find which type of feature was held in the array. As an example, if the current feature in the array contained the property `riverid` it could be determined to be a river feature and stored in the appropriate state variable. By using this method the user could select what data would be displayed on the popup by controlling which layers were active on the map when the click was performed.

The data from the active features that overlapped with the click bounds was then displayed in the popup at the interaction location. In addition, the popup contained the coordinates, the yearly average wind speed, and yearly average solar potential of the location. However, problems with the method of acquiring data was quickly realized.

4.8 Bounds and missing data problem.

The first problem with this method was that in most cases, when the user selected a location there would not be any overlapping features, leading to only wind, solar, township and district data being displayed. As CORE-KIT is further developed, this would lead to problems with calculating demand if no settlement data was gathered, or calculating hydro potential if no river data was gathered.

The second problem was not as obvious. As polyline and polygon features have large bounds, it could cause multiple features of the same type to overlap with the click bounds. This caused a problem with selecting correct features, as CORE-KIT currently is capable of displaying data from a single feature of each type. When this problem occurred, the feature which was added last in to the overlapping features array would always be selected. In case of polygon features, the problem was easy to detect and a workaround was implemented

using a database query. Listing 3 shows how the correct polygon feature is acquired by comparing the location the map was clicked on with the district features in the database using an SQL query.

```
SELECT *
FROM public.districts_shan_south
WHERE ST_Within(ST_SetSRID(ST_Point($1, $2),0), geom::geometry);
```

Listing 3. SQL query to get the district feature that contains the map interaction location. The longitude replaces the “\$1”, and latitude replaces the “\$2”. These values are passed to the query as parameters.

Figure 5 shows the end result of a map click with multiple overlays active and multiple overlapping features present at the location.



Figure 5. Shows the end result of the map click event using the first implemented method for map interaction.

When multiple polyline features overlapped it was much harder to detect, as at the time the polyline features did not yet have the mouse over function tied to them. Once this problem was noticed it led to the implementation of the second method for selecting features.

4.9 Second Method: SQL queries with spatial functions

The second method solves both of the issues found with the first method. Instead of creating click bounds and comparing them to features active on the map the second method relies only on SQL queries using spatial functions. Data from the polygon features are gathered the same way as in the first method. The other vector features are acquired by utilizing nearest neighbor search.

Nearest neighbor search is used to find nearest candidate features to the query feature. The nearest features are found by comparing the distance of each candidate feature to the query feature. In case of CORE-KIT, the nearest settlement, river, and medium voltage grid features can be found by using the location where the map click occurred as the query feature. Listing 4 shows an example of how the nearest neighbor search can be performed.

```
SELECT*,
ST_Transform(public.rivers.geom, 26918),
public.rivers.geom < - > ST_SetSRID(ST_Point($1, $2), 26918)::geometry
AS dist
FROM public.rivers
ORDER BY dist
LIMIT 1
```

Listing 4. Shows the SQL query for retrieving the nearest river feature to the selected location on the map.

The ST_Transform function returns the geometry of each feature compliant to a specific spatial reference. In the example above 26918 is used, as the vector features in the database were added using that spatial reference system. The ST_Point function creates a point type feature with coordinates of the location the map interaction occurred on. The ST_SetSRID function defines the spatial reference system used for the point. It is imperative that the SRID of the candidate and query features match for the proximity function to work.

The point is then compared with all the candidate features and ordered by their distance to the point. By limiting the results to a single feature, only the data from the nearest feature is received.

In addition to finding the nearest feature, the distance to the nearest feature can be acquired through the same query. In its current form 2D Cartesian (planar) distance is used, but it could be possible to convert the distance into meters.

The popup generated when the map was clicked was also changed. It now always displays the available data regardless of which overlays were active when the interaction was performed. In addition, the distance to the nearest settlement, river segment and medium voltage grid was added. A marker was also added to the location that was clicked. The popup is now tied to the marker. This allows the ability to close and reopen the popup without the need to trigger another map click event. The end result of a map click interaction, including solar potential graphs can be seen in Figure 6.

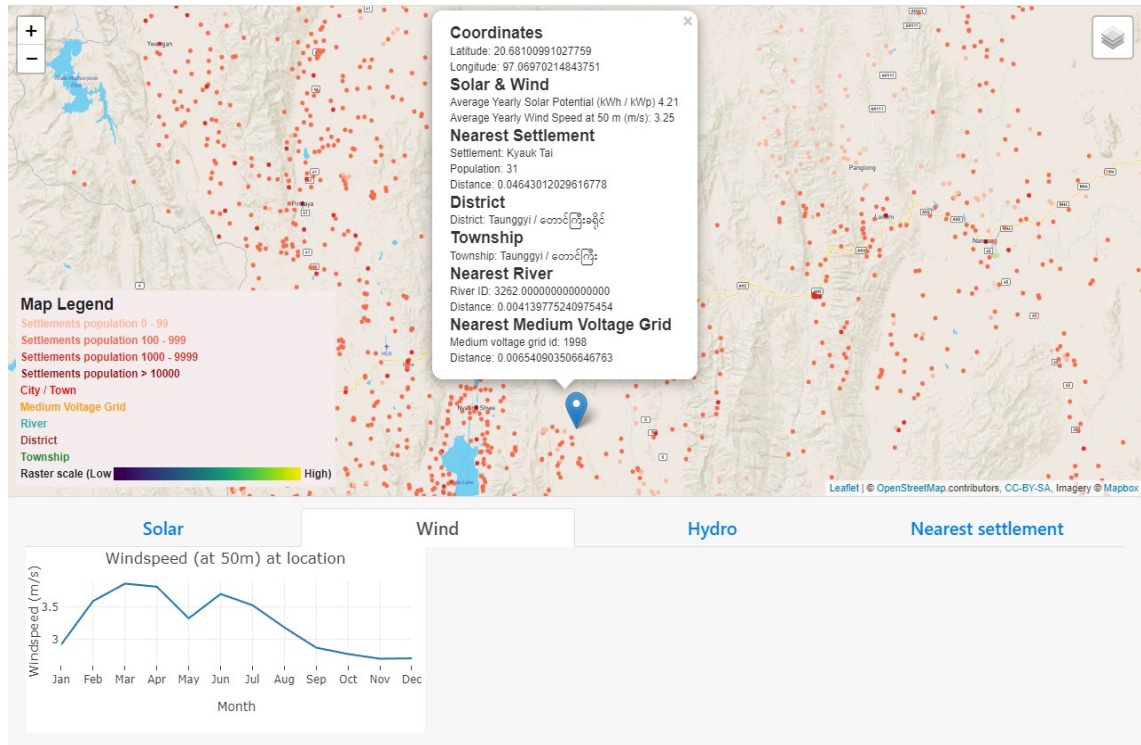


Figure 6. Shows the result of a map click event with just the settlement overlay active using the second method for acquiring relevant features.

As Figure 6 above shows the popup contains the limited information from each relevant feature. In addition the monthly average wind speed at 50 meters graph can be seen on the information tabs below the map.

5 Future development

5.1 User interface

The prototype's interface can be improved in multiple ways. The most pressing of these would be custom layer controls and highlighting the features that data were gathered from. In addition, attributions for the data used CORE-KIT must be made available before the prototype could be publicly hosted. Another area for improvement could be the map legend. The legend could include only features that are active on the map and numerical values given for the raster scale.

5.2 Rasters

As for the rasters, future developments would be to only allow a single raster overlay to be active at once. Having multiple rasters active makes them muddled and impossible to read properly. In addition, rasters for each month individually could be made available for both wind speed and solar potential.

A raster of each specific month could be added using the same method as used in the currently available rasters, but instead of calculating the mean value of the twelve bands, a value contained in a specific band would be used instead.

5.3 Estimating demand and electricity generation

As the development of the CORE-KIT continues, estimating demand will be added. The demand can be estimated by using the number of households in a settlement multiplied by a power usage per household value set by the user. Baseline value could also be included based on a study conducted by Smart Power Myanmar. The study concluded that average daily electricity

consumption of a rural household was 0.67 kilowatt hours per day. [25.] The daily load curve shown in the study could also be roughly replicated.

In the future, CORE-KIT could also estimate electricity generation by using solar and wind. The generation using solar can be estimated by allowing the user to input the total panel capacity, measured in kilowatt hours. The rough generation estimate could then be calculated by multiplying the total panel capacity with the kilowatt hour per kilowatt peak.

Alternatively, CORE-KIT could calculate the total panel capacity and size of the battery required to meet the demand completely. In this case not only the peak demand would have to be met, but have large enough charge stored in the battery to meet night time demand, as the electricity generation during night time would be minimal if any.

A rough estimate of wind based electricity generation could be achieved by allowing the user to input information about the wind turbine. The inputs required would be number of units, turbine coefficient percentage, rotor blade length in meters and cut-in wind speed in meters per second. Using these values the rough estimation of electricity generated could be calculated using the following formula.

$$P_{tp} = \frac{1}{2} \rho A v^3 C_p$$

In the equation P_{tp} refers to potential theoretical power. ρ refers to constant air density at sea level at 15°C. A refers to the sweep area of the rotor, v^3 refers to wind speed and C_p refers to the power coefficient. [20, p. 26-27.] The result of the equation is in watts, to convert it to kilowatt hours generated during a day the result will be divided by 1000 to convert it to kilowatts and then multiplied by 24 hours. Listing 5 shows how to calculate daily electricity generation for each month using the formula above in JavaScript.

```

var blade = 2.5;
var airDensity = 1.225;
var cp = 0.35;
var cutInSpeed = 3.5;
var units = 2;
var rotorSweepArea = Math.PI * (blade * blade);
for(var i = 0; i < 12; i++){
    if(windPtp[i] >= cutInSpeed){
        windGen.push((0.5 * airDensity * rotorSweepArea * (windPtp[i]
            * windPtp[i] * windPtp[i])* cp) / 1000 * 24 * units);
    }else{
        windGen.push(0);
    }
}

```

Listing 5. shows how the calculations to estimate daily wind generation for each month could be implemented.

In the code above (see listing 4, Wind generation calculation) the windPtp refers to an array containing the average wind speed of each month. The windGen array contains the estimated daily electricity production for each month in kilowatt hours.

5.4 Micro-hydro

Estimating electricity generation using micro-hydro provides additional challenges. Monthly river discharge data modelled with Hydrostreamer could be added to the database. Hydrostreamer is an open source R software library, which can be used to convert existing runoff products to desirable spatial units, without requiring extensive hydrological modelling knowledge. Hydrostreamer handles interpolation, river routing and data assimilation. [26.]

The data modeled with Hydrostreamer would then be presented as monthly discharge and as flow-duration curve. However this alone would not be enough for estimating kinetic micro-hydro potential, as these systems rely on elevation change to generate kinetic energy.

To estimate kinetic micro-hydro the user should be allowed to select an intake location, powerhouse location and create a pipeline between them with a defined elevation drop near the powerhouse to generate kinetic energy.

Optionally this could be simplified by using the discharge and flow-duration curve of the nearest river segment and allowing the user to input the elevation change between the intake and powerhouse.

5.5 Expanding coverage and uniform spatial reference system

Early in the prototype's development it was decided to focus on a specific portion of Myanmar, namely the southern Shan State. In the future, CORE-KIT could expand to include the entire Shan State or even the whole country. As data covering the whole country would be available. Currently CORE-KIT uses two different spatial reference systems. 4326 for rasters and 26918 for vector features. This does not cause any issues as is, however it would be beneficial to use a single system for consistency. This could be changed when expanding the coverage of CORE-KIT, as all data would have to be imported to the database again regardless.

6 Conclusion

The goal of this thesis was to create an interactive web application to map renewable energy resources (wind and solar) and demand points (settlements) in southern Shan State, Myanmar. The study contained three main objectives; documenting and storing datasets, gathering and presenting data, as well as identifying features to improve and add during future development cycles.

The first object was achieved by documenting the source of each dataset used in CORE-KIT. In addition if modifications were made to the source datasets, the modifications were documented as well. The study demonstrated how vector and raster data was imported into a PostGIS enabled PostgreSQL database.

For the second objective two methods were implemented. The first method compared bounds of active features on the map to bounds created for the map click event. By comparing the bounds, an array of overlapping features was established. The features were selected and saved as state variables by either directly through the array or by querying the database. However, notable issues were found using this method. These problems were corrected by the second method, which involved using nearest neighbor search to find relevant features. The data contained in those features was then presented inside a popup in limited form and in the info tabs in more detail.

The third objective was achieved by documenting areas of the application that require further development to meet the goals of CORE-KIT, or before the application could be hosted on the internet. Examples of how these could be added or improved were also suggested.

References

- 1 Alliance for Rural Electrification - Benefits of Clean Rural Electrification [Internet]. Alliance for Rural Electrification. URL: <https://www.ruralelec.org/benefits-clean-rural-electrification> Accessed 2 May 2022.
- 2 Khine Wah Lwin. Burma Country Commercial Guide [Internet]. International Trade Administration; 2021. URL: <https://www.trade.gov/country-commercial-guides/burma-energy> Accessed 5 November 2021.
- 3 Ministry of Electricity and Energy - Status of Current Progress & Future Plan of National Electrification Project (NEP) [Internet]. Ministry of Electricity and Energy; 2022. URL: <https://www.moe.gov.mm/en/ignite/contentView/2214> Accessed 1 May 2022.
- 4 Tarek Ketelsen. [Interview]. 4 November 2021.
- 5 Caitlin Dempsey. What is GIS? [Internet]. GIS Lounge; 2021. URL: <https://www.gislounge.com/what-is-gis/> Accessed 2 November 2021.
- 6 Sutton T, Dassau O, Sutton M. A gentle introduction to GIS – Vector Data [Internet]. QGIS Documentation; 2021. URL: https://docs.qgis.org/3.22/en/docs/gentle_gis_introduction/vector_data.html Accessed 2 November 2021.
- 7 Vitaly Rudnytskiy. Understanding Z and M coordinates in SAP HANA Spatial [Internet]. SAP Developer Center; 2021; URL: <https://developers.sap.com/tutorials/hana-spatial-intro5-z-m-coordinates.html> Accessed 8 November 2021.
- 8 Sutton T, Dassau O, Sutton M. 2009. A gentle introduction to GIS – Raster Data [Internet] QGIS Documentation; 2021. URL: https://docs.qgis.org/3.22/en/docs/gentle_gis_introduction/raster_data.html Accessed 6 November 2021.
- 9 Caitlin Dempsey. Types of GIS data explored: Vector and Raster [Internet]. GIS Lounge; 2021. URL: <https://www.gislounge.com/geodatabases-explored-vector-and-raster-data/> Accessed 7 November 2021.
- 10 Tom MacWright. More than you ever wanted to know about GeoJSON [Internet]. Macwright.com; 2015. URL: <https://macwright.org/2015/03/23/geojson-second-bite> Accessed 1 November 2021.

- 11 The Internet Engineering Task Force. Rfc7946 Internet Standards Track document [Internet]. The Internet Engineering Task Force; 2016. URL: <https://datatracker.ietf.org/doc/html/rfc7946> Accessed 1 November 2021.
- 12 Caitlin Dempsey. What is a GeoTIFF? [Internet]. GIS Lounge; 2019. URL: <https://www.gislounge.com/what-is-a-geotiff/> Accessed 3 May 2022.
- 13 Elmasri R, Navathe SB. Fundamentals of Database Systems [e-book]. Pearson; 2015. URL: <http://103.38.12.142:8081/jspui/bitstream/123456789/507/1/Fundamentals%20of%20Database%20Systems%2C.%20Ramez%20Elmasri%20and%20Shamkant%20B.%20Navathe.pdf> Accessed 9 November 2021.
- 14 Kristi Castro 2018. Object-oriented Data Model [Internet]. Tutorials Point; 2018. URL: <https://www.tutorialspoint.com/Object-oriented-Data-Model> Accessed 1 December 2021.
- 15 PostgreSQL – About [Internet]. PostgreSQL; 2021. URL: <https://www.postgresql.org/about/> Accessed 11 November 2021.
- 16 Kanukov AS, Ivanov PG. Geological information database integration into a geographic information modeling system [PDF]. IOP Science; 2021. URL: <https://iopscience.iop.org/article/10.1088/1757-899X/1083/1/012085/pdf> Accessed 15 November 2021.
- 17 QGIS - QGIS Readme.md [Internet] Github; 2021. URL: <https://github.com/qgis/QGIS> Accessed 22 November 2021.
- 18 OpenStreetMap-carto – Cartography.md [Internet]. Github; 2019. URL: <https://github.com/gravitystorm/openstreetmap-carto/blob/master/CARTOGRAPHY.md> Accessed 19 November 2021.
- 19 Energydata.info - Myanmar - Populated Settlements Data [Internet]. Energydata.info; 2018. URL: <https://energydata.info/dataset/myanmar-populated-settlements-data> Accessed 21 November 2021.
- 20 Andrew Said. Myanmar's renewable energy potential - Open source resources mapping for electrification of rural communities [PDF]. Aalto University; 2018. URL: https://aalto.doc.aalto.fi/bitstream/handle/123456789/31609/master_Said_Andrew_2018.pdf?sequence=1&isAllowed=y Accessed 30 November 2021.

- 21 PostGIS. PostGIS 3.2.2dev Manual – Chapter 11. Raster Data Management, Queries, and Applications [Internet]. PostGIS; 2022. URL: https://postgis.net/docs/using_raster_dataman.html Accessed 3 May 2022.
- 22 Leaflet - Leaflet documentation [Internet]. Leaflet; 2021. URL: <https://leafletjs.com/reference.html> Accessed 19 November 2021.
- 23 React Leaflet - React Leaflet documentation [Internet]. React-leaflet; 2021. URL: <https://react-leaflet.js.org/docs/start-introduction/> Accessed 24 November 2021.
- 24 GeoTIFF/georaster-for-leaflet – Readme.md [Internet]. Github; 2021. URL: <https://github.com/GeoTIFF/georaster-layer-for-leaflet> Accessed 27 November 2021.
- 25 Smart Power Myanmar. Technical brief consumption trends [PDF]. Smart Power Myanmar; 2020. URL: <https://smartpowermyanmar.org/wp-content/uploads/2020/10/Technical-Brief-Consumption-Trends-1.pdf> Accessed 30 November 2021.
- 26 Kallio M, Guillaume JHA, Virkki V, Kumm M, Virrantaus K. Hydrostreamer v1.0 – improved streamflow predictions for local applications from an ensemble of downscaled global runoff products [Internet]. European Geosciences Union; 2021. URL: <https://gmd.copernicus.org/articles/14/5155/2021/> Accessed 2 December 2021.