Energy Prediction Digital Solution

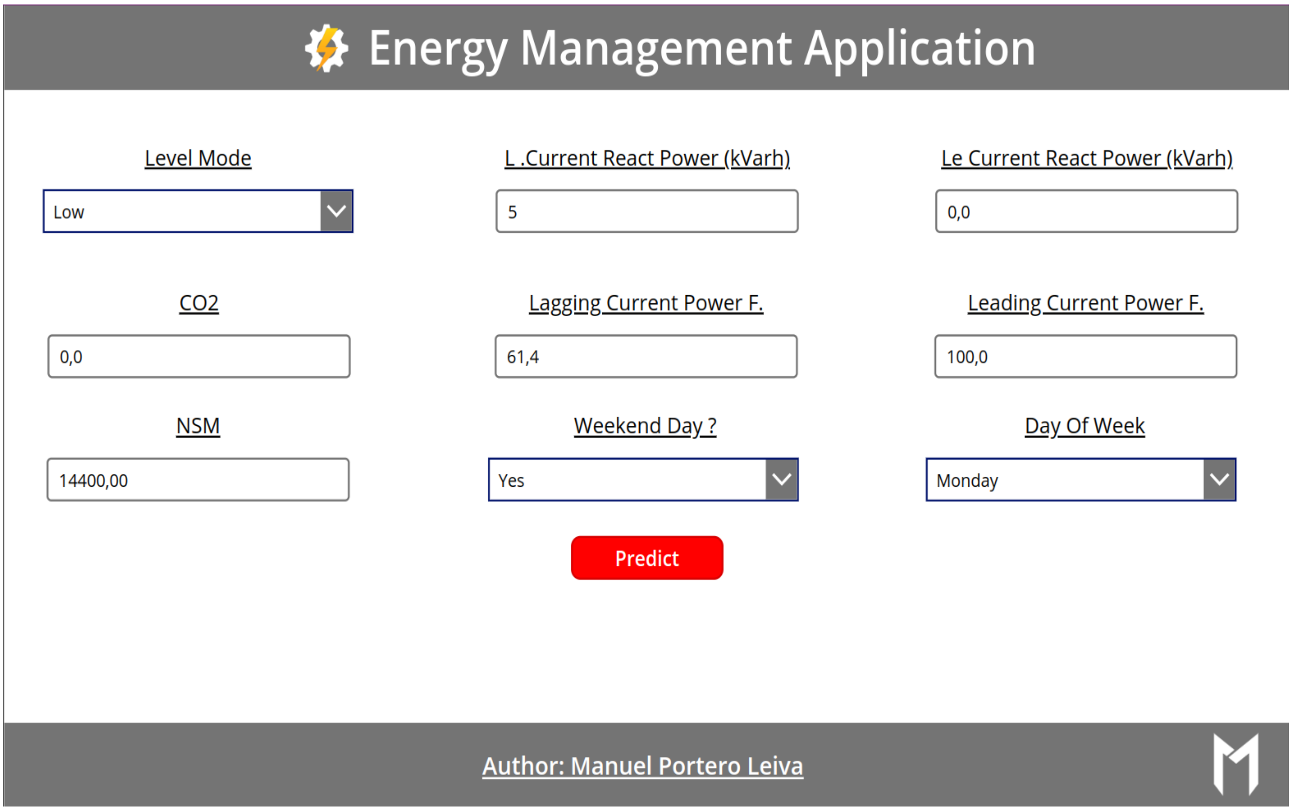
Documentation

20/07/2024

Author: Manuel Portero Leiva

Introduction

This document has the purpose to explain the different parts of the Energy Prediction Digital solution, it's code and functionalities, for understanding and replication purposes. The different parts of the architecture solution are show below.



Energy Management Application

Level Mode	L .Current React Power (kVarh)	Le Current React Power (kVarh)
Low	5	0,0
CO2	Lagging Current Power F.	Leading Current Power F.
0,0	61,4	100,0
NSM	Weekend Day ?	Day Of Week
14400,00	Yes	Monday

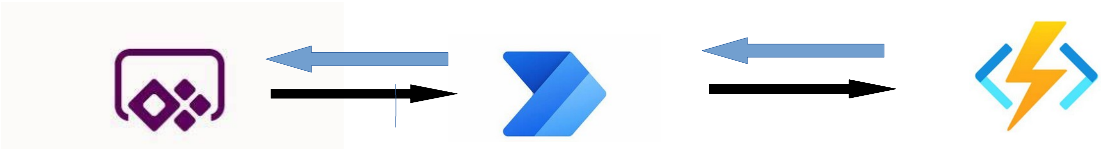
Author: Manuel Portero Leiva

Picture 1: Energy Prediction App Layout

Architecture

The composition of the architecture starts in the PowerApp. Once an equipment design is choosen an equipment design screen is shown, the user modify the equipment and after change the equipment parameters all the process is recalculared via PowerAutomate / Function app.

A full diagram of the solution is shown below.

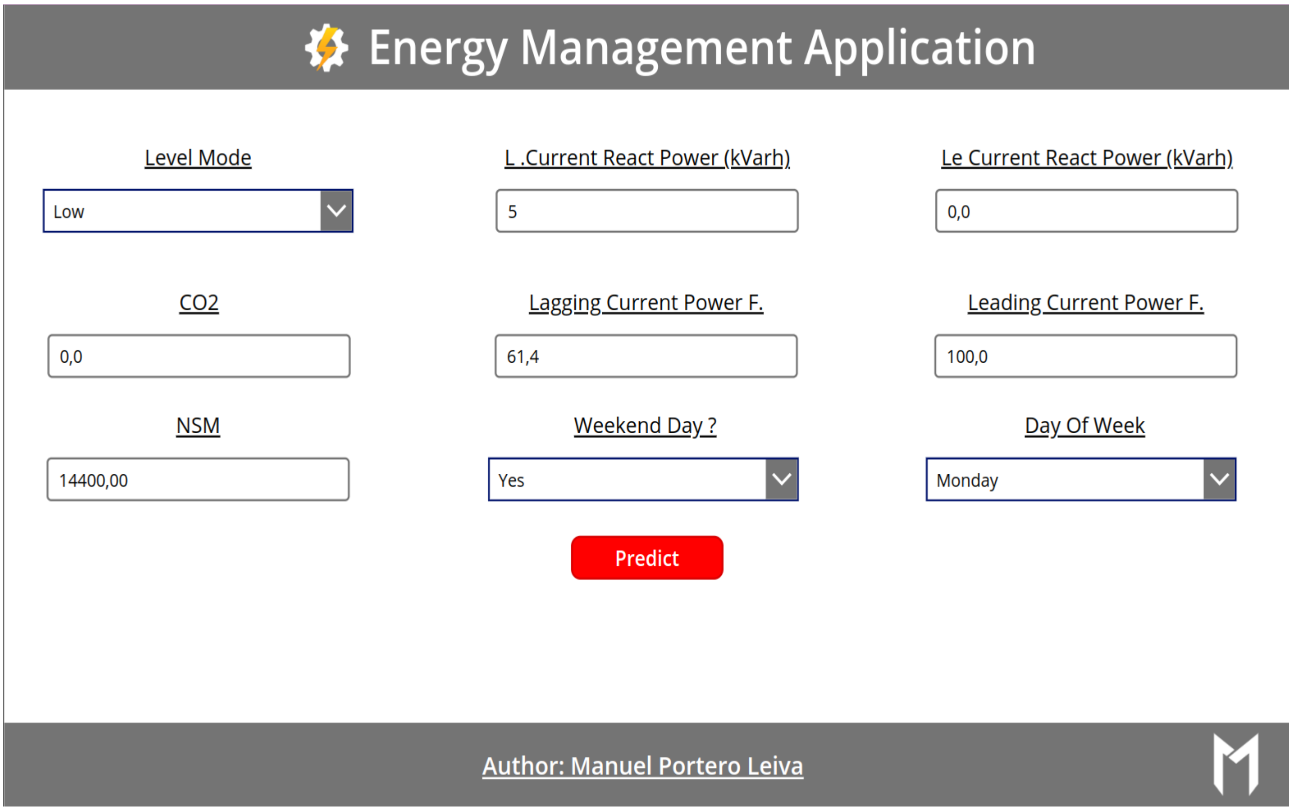


Picture 2: Energy Prediction Architecture

PowerApp

Home Screen

The home screen is composed by the predicton fields and a prediction button.



Energy Management Application

Level Mode: Low

L .Current React Power (kVarh): 5

Le Current React Power (kVarh): 0,0

CO2: 0,0

Lagging Current Power F.: 61,4

Leading Current Power F.: 100,0

NSM: 14400,00

Weekend Day ?: Yes

Day Of Week: Monday

Picture 4: S Beer Process Simulation main Screen

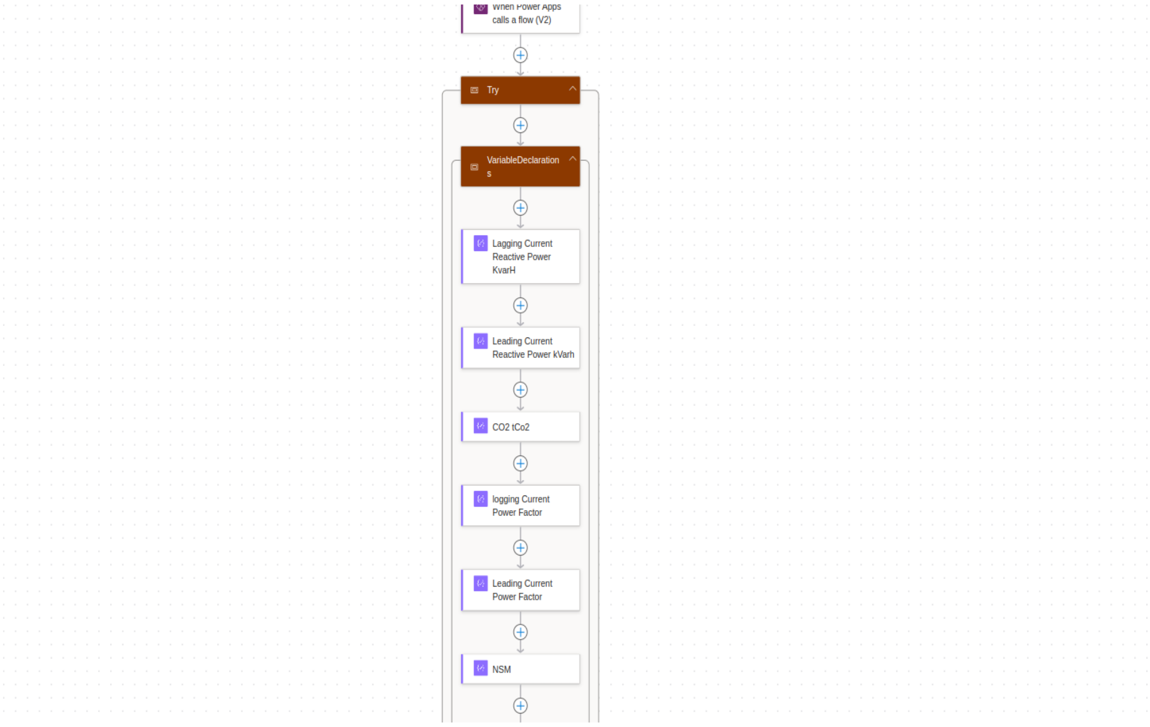
The home screen full code is shown below :

Predict button.

```
If(
Set(
UsageValue;
EnergyPredictionAutomation.Run(
Value(LCurrentReactPower_Input.Text);
Value(LeCurrentReactPower_Input.Text);
Value(Co2_Input.Text);
Value(LaggingCurrentPowerF_Input.Text);
Value(LeadingCurrentPowerF_Input.Text);
Value(NSM_Input.Text);
weekendDay;
dayOfWeek;
LevelMode
).usage_kw
);
Set(
usageVisible;
true
);;
Notify(
"Prediction executed successfully!!";
NotificationType.Success
);
Notify(
"Error with prediction execution...";
NotificationType.Error
)
)
```

PowerAutomate Flow

The PowerAutomate flow receive the paremeters from the PowerApp, call the Azure function with an Http Request action and update and return the consumption result parameter to the powerApp.


When Power Apps calls a flow (V2)
Try
VariableDeclarations
Lagging Current Reactive Power KvarH
Leading Current Reactive Power kVarh
CO2 tCo2
logging Current Power Factor
Leading Current Power Factor
NSM

Picture 6: Energy prediction flow sample

Azure Function

The energy prediction azure function will receive the parameters from the PowerAutomate flow and will calculate the energy consumption. The code of each azure functions are shown below :

Energy Prediction Azure Function:

```
import azure.functions as func
import logging
import pickle
from datetime import datetime

app = func.FunctionApp(http_auth_level=func.AuthLevel.ANONYMOUS)

@app.route(route="aFunctionEnergyPrediction")
def aFunctionEnergyPrediction(req: func.HttpRequest) -> func.HttpResponse:

    logging.info('Starting prediction ....')


    try:
        req_body = req.get_json()
    except ValueError:
        pass
    else:

        Lagging_Current_Reactive_Power_KvarH =
req_body.get('Lagging_Current_Reactive_Power_KvarH')
        Leading_Current_Reactive_Power_kVarh =
req_body.get('Leading_Current_Reactive_Power_kVarh')
        CO2_tCo2 = req_body.get('CO2_tCo2')
        logging_Current_Power_Factor =
req_body.get('logging_Current_Power_Factor')
        Leading_Current_Power_Factor =
req_body.get('Leading_Current_Power_Factor')
        NSM = req_body.get('NSM')
        WeekStatus = req_body.get('WeekStatus')
        Day_of_week = req_body.get('Day_of_week')
        Load_Type = req_body.get('Load_Type')

        X_test = [Lagging_Current_Reactive_Power_KvarH,
Leading_Current_Reactive_Power_kVarh, CO2_tCo2,
logging_Current_Power_Factor, Leading_Current_Power_Factor, NSM,
WeekStatus, Day_of_week, Load_Type]

        # Loading model

        model_pkl_file = "./energyPredictorModel.pkl"
        with open(model_pkl_file, 'rb') as file :
            model = pickle.load(file)

        Usage_Kw =  model.predict([X_test])
        logging.info(f"Usage_Kw : {Usage_Kw}")
        return func.HttpResponse(str(Usage_Kw), status_code=200)
```

Machine Learning model

The development of the machine learning model and conclusions are discussed in the related document “EnergyConsumptionSteelFactories.pdf”

Solution code :

Solution code : Github repository link