Pseudo Code

```
BEGIN Smart_Irrigation_System
    INITIALIZE microcontroller
    INITIALIZE GSM_module
    INITIALIZE sensors

    WHILE true DO
        soil_moisture = READ(soil_moisture_sensor)
        temperature = READ(temperature_sensor)
        humidity = READ(humidity_sensor)
        IF user_requests_data THEN
          SEND_SMS(user, "Moisture: " + soil_moisture + ",Temp:"
          +temperature + ", Humidity: " + humidity)
        ENDIF

        IF soil_moisture < threshold THEN
         ACTIVATE irrigation_system
         WAIT(duration)
         WHILE irrigation_system_active & soil_moisture<optimal_level
         DO
             CONTINUE irrigation
         ENDWHILE
         DEACTIVATE irrigation_system
         SEND_SMS(user,"Irrigation done.Current moisture:"+ soil_moisture)
        ENDIF
        IF receive_user_command THEN
            PARSE command
            UPDATE system_parameters
        ENDIF

        WAIT(interval)  // wait before the next reading
    ENDWHILE
END Smart_Irrigation_System
```

```
RandomForestClassifier(n_estimators=500,
max_depth=20, max_leaf_nodes=10000)
custom_random_forest_classifier.fit(data_train,labels_train)
custom_classifier_prediction_label =
custom_random_forest_classifier.predict(data_test)
confusionMatrix2 =
confusion_matrix(labels_test,custom_classifier_prediction_label)
print(confusionMatrix2)
accuracy_score(labels_test,custom_classifier_prediction_label)
import matplotlib.pyplot as plt
import numpy as np

importances (the higher, the more important the feature).

in the order the features were fed to the algorithm
importances = custom_random_forest_classifier.feature_importances_
#std = np.std([tree.feature_importances_ for tree in
custom_random_forest_classifier.estimators_],axis=0)

important.
indices = np.argsort(importances)[::-1]
print(f"indices of columns : {indices}")
# Print the feature ranking
print("\n ***Feature ranking: *** \n")
print("Feature name : Importance")
for f in range(data_train.shape[1]):
print(f"{f+1} {data_train.columns[indices[f]]} :
{importances[indices[f]]} \n")
print(" The blue bars are the feature importances of the
randomforest classifier,
along with their inter-trees variability*")
# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(data_train.shape[1]), importances[indices],
color="b", align="center")
are not making it plot.
plt.xticks(range(data_train.shape[1]), data_train.columns[indices])
plt.xlim([-1, data_train.shape[1]])
plt.rcParams['figure.figsize'] = (35,15)
#this will increase the size of the plot
plt.show()
```