CS 308 Project

Siri (Smart Irrigation)

Dheeraj 120050061
Nishanth 120050064
Nikhil 120050070
Mahindar 120050073

Problem Statement

The aim of our project is to minimize the usage of water.

Currently in many of the smart irrigation farms water to the plants are controlled based on timers. This may lead to unnecessary wastage of water. We use the factors like soil moisture, temperature and water level to determine the requirement for the need of water by the plant

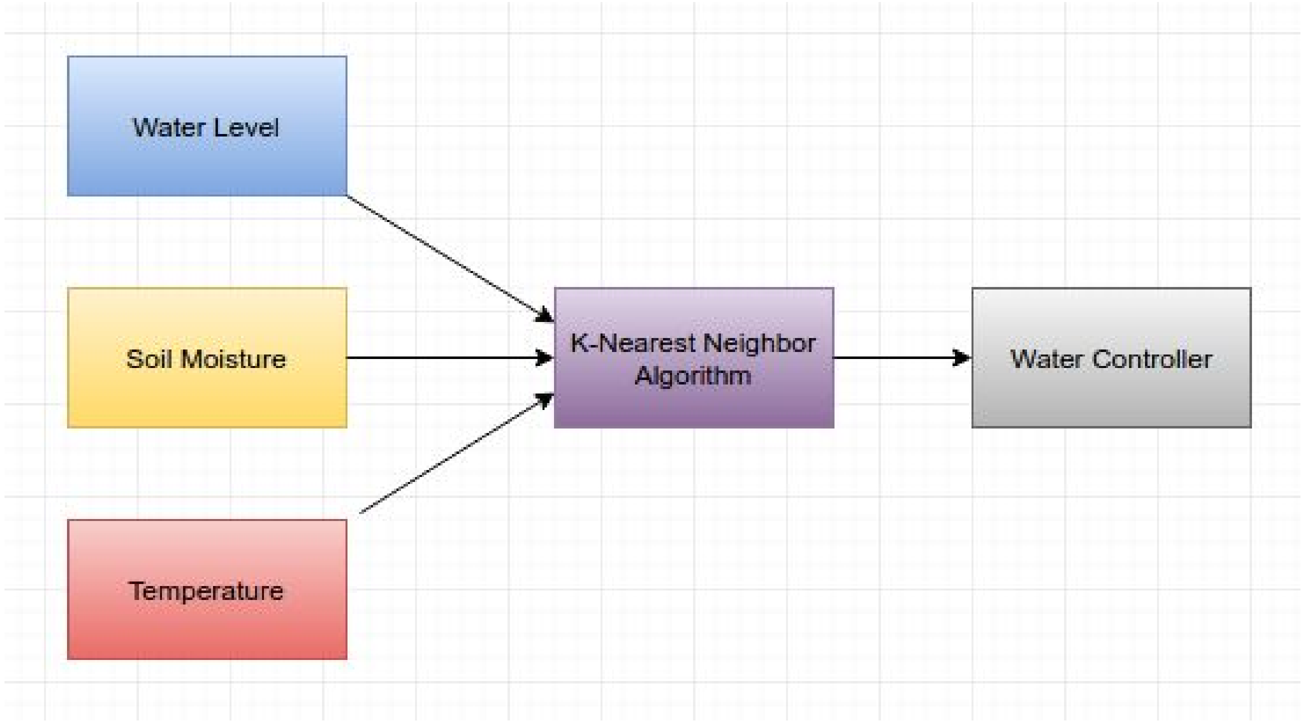
Requirements and Task Specifications

- Our project requires a LM393 soil moisture sensor, TIVA CC3100 Boosterpack, a water level sensor and TIVA C Series TM4C123G microcontroller which has an in-built temperature sensor.
- The tasks of our projects are the following:
    a. Learning and testing the working of the sensors
    b. Coming up with an Machine Learning algorithm to understand the requirement of water supply when required and implementing it
    c. Learn and implementing a wifi server using the Boosterpack which sends a HTML page on demand displaying the current water status, soil moisture and temperature
    d. Writing a script which can be used in a cron job to maintain log of the data in the HTML page sent by the server

Project Plan

- We first executed the task a followed by b and then executed tasks c and d in parallel
- Tasks a and b were done with a combined effort by all of us where some us coded while some read the documentation and ML algorithms and the tasks c and d were done by pair of us.
- Dates of Task completion
  - Task a - 23/3/16
  - Task b - 30/3/16
  - Task c - 3/4/16
  - Task d - 3/4/16

Workflow diagram



Workflow Description

- Soil moisture, temperature and water empty state values are collected from the respective sensors
- The K-nearest neighbour algorithm takes these values as input and then calculates the K-nearest neighbors
- Now the depending on the majority of these K neighbors we decide on requirement of the water
- Depending on the decision either red or green LED

Innovation and Challenges

Challenge:
Initially we planned to use the ultrasonic sensor to measure the exact water level and then use regression to fit a polynomial of 1 or 2 degree over the data of water level, temperature, soil moisture. But due to the inaccuracy of the ultrasonic sensor for less depths of water we decided to use a water level sensor which gives low value in absence of water.
Solution:
We then developed a modification of the k-nearest neighbour algorithm with a voting mechanism which predicts whether there is a need for water level. This algorithm needs less training data when compared to regression model and with just 45 data points we were able to predict at high accuracies

Challenge:
Generation of data points for the k-nearest neighbor algorithm
Solution:
We divided the soil moisture and temperature into 3 parts of low, medium and high. Then depending on the region we decide on whether to release water or not.

Tasks Completed

Task a: Learning and testing the working of the sensors

Problem: The ultrasonic depth measurement sensor failed at low depth values

Solution: We used a new water level sensor which gives low value on water container being empty.

We collected 5 samples from this sensor and depending on the median low or higher value is decided

Task b: Coming up with an Machine Learning algorithm to understand the requirement of water supply when required and implementing it

Problem: Shifting from regression to a classification problem

Solution: Using a modified version of KNN along with a voting mechanism among the k-nearest neighbors

Tasks Completed

Task c: Learn and implementing a wifi server using the Boosterpack which sends a HTML page on demand displaying the current water status, soil moisture and temperature

Problem: Learning the usage of the creating a wifi server and sending html page

Solution: Learnt from http://energia.nu/pin-maps/guide_cc3100boosterpack/

Task d: Writing a script which can be used in a cron job to maintain log of the data in the HTML page sent by the server

Problem: Maintaining a log of the work of the decisions of the system

Solution: Used a bash script as a cron job which gets the HTML page from the server and parses it stores the values in a html log file

Test Cases

- Below are the 9 test cases considered for our testing where we assume water is low:
- {L, L, 1}, {L, M, 0}, {L, H, 0}, {M, L, 1}, {M, M, 0}, {M, H, 0}, {H, L, 1}, {H, M, 1}, {H, H, 0} where L is low, M is medium and H is high and in {X, Y, Z} X corresponds to temperature, Y to soil moisture and Z = 0 implies no water needed and Z = 1 implies water is required
- For example {L, H, 0} means when the temperature is low and there is high soil moisture no water is required since soil has sufficient water. So the decision is 0 for water requirement
- Another test is printing the current temperature, soil moisture and water level state and then checking the same on the HTML page we obtain from the sensor

Performance Metrics

- The water level sensor produces some inaccurate outliers during measurement. We calculated 10 values from the sensor and use the median to collect data from it. Since a misinterpreted reading of the sensor may cause overflow/wastage of water and hence lead to wrong decisions
- The delay between the consecutive readings/decisions affects the battery life of the system and since these measurements differ at a less pace with respect to time we can increase the delay
- Connection between the wifi module and the system which saves the logs is important for the storage of the decisions and data collected by the TIVA board

Re-usability features

- Wrote module make_decision which implements the KNN algorithm and the voting mechanism
- The wifi set-up and the server based code is reusable and can be used to support a number of pages
- Wrote modules to read the pin values and can be used for multiple sensors
-

Future Enhancements

- Current system supports the decision making for 2-3 plants in a smaller area. It can be extended to a large area by having wireless sensors sending data to a board where it will be stored in a queue on which decisions will be made
- Introducing a queue like buffer to store the readings so that if there is a break in the connection, the readings can be stored in it
- Using other factors of the weather to improve further decisions like moisture in atmosphere to determine whether there will be rain or not