THESIS

IMPROVING ACCURACY FOR SUGAR BEET AND DEVELOPING AN iOS APP TO INCREASE FUNCTIONALITY OF A COLORADO IRRIGATION SCHEDULER

Submitted by

Andrew Charles Bartlett

Department of Soil and Crop Sciences

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2014

Master's Committee:

Advisor: Allan A. Andales

Troy Bauder
Mazdak Arabi

ABSTRACT

IMPROVING ACCURACY FOR SUGAR BEET AND DEVELOPING AN iOS APP TO INCREASE FUNCTIONALITY OF A COLORADO IRRIGATION SCHEDULER

Modeling actual crop water usage allows for improved information-based decision making and ultimately more effective use of water allocations within irrigated agriculture. Evapotranspiration (ET), a dynamic process of water loss through the soil surface (evaporation) and plant stomata (transpiration), is the main component of consumptive water use. Scheduling agricultural irrigation events is an effective tactic to minimize crop stress while avoiding unnecessary irrigation. A cloud based irrigation scheduling tool (WISE – Water Irrigation Scheduling for Efficient Application) which applies the soil water balance (SWB) approach, has been developed on the eRAMS (Environmental Risk Assessment Management System) platform. Actual crop water usage (ET_c) is the main cause of the depletion of soil moisture, thus ET_c is one of the most important variables within the SWB. Multiple ET equations have been developed as a function of a handful of meteorological measurements including the equation used in this thesis, the 2005 American Society of Civil Engineers (ASCE-EWRI) Standardized Reference equation. The alfalfa based reference evapotranspiration (ET_r) models the water loss via ET for a 0.5 m tall, well watered alfalfa stand. In order to model sugar beet water use, an empirically derived crop coefficient (K_{cr}) curve is applied to the alfalfa reference ($ET_c = ET_r \times K_{cr}$). Region specific sugar beet crop coefficient values are available; however, these values have not yet been adjusted for the semi-arid climate of Northeastern Colorado.

The first objective of this thesis was to modify the sugar beet K_{cr} curve for the semi-arid climate of Northeastern Colorado to increase the accuracy of sugar beet scheduling within WISE. By using the soil water balance and observing plant growth and water uptake rates, it was discovered that the original coefficient was drastically overestimating ET_c. Shortening the full canopy stage by delaying the initial point (cutoff 2) from 33% to 43% maturity and reducing the length until senescence from 83% to 69% maturity reduced predicted water use to an acceptable value. After comparing actual soil water deficits (D) with modeled values for both the original and adjusted K_{cr} over two growing seasons, it was found that the relative error (RE) of daily D over all fields was decreased from RE values ranging from 11% - 300% down to RE values ranging from 0% - 265%. Large errors were caused by uncertainties in soil properties, effects of hail damage on actual leaf area and ET_c, spatial variability in precipitation or irrigation, and differences in field micro-climate and measured weather station data.

The second objective of this thesis was to describe the development and purpose of an iPhone and iPad application that was created to add mobile functionality to the WISE tool. This app allows users to view their field's current soil moisture profile, previous day's weather, upload irrigation and precipitation amounts, and calculate gross irrigation amounts as a function of flow rate, length of application, and acreage. The new sugar beet K_{cr} curve and the iOS app can lead to more effective irrigation scheduling in agriculture within Colorado.

ACKNOWLEDGEMENTS

I cannot image where I would be today without trust in my eternal savior, Jesus Christ. Sometimes science tries to create explanations for unexplainable processes; however, the further I delve into science, the more I see His delicate creation and design. His grace and mercy provide strength and meaning to the work that I have accomplished, and will continue to do so. “I can do all things through Him who strengthens me,” Philippians 4:13. Throughout my life, my family, Charlie, Patty, and Stacey, have been by my side, encouraging me to do my best and to strive for the top. The practical lessons that I learned growing up in a rural setting, have been a vital part to my academic success. The professors at Northeastern Junior College and Colorado State University have instilled a passion inside me for soil and crop sciences. Dr. Allan Andales took a risk with me as a young undergraduate with no field research experience or computer programming background. His guidance and trust have been extremely appreciated. My committee members, Dr. Mazdak Arabi, and Troy Bauder, provided invaluable insight and direction so I could get the most out of this degree. During my two summers of research at CSU, Erik Wardle and Joel Schneekloth invested numerous hours in assisting me with field research and other project tasks. Kyle Traff, the genius programmer behind the irrigation scheduler, also provided countless hours of assistance with the development of the iPhone app. Finally, I would not be here today, writing this thesis, if it was not for my wife, Meagan. She embraced my love for learning and recognized the potential that I had as a graduate student. She always pushes me to accomplish tasks I first distinguish unimaginable. I thank the Lord every day that He put her by my side. We are only a few years into our marriage but I am already aware of the fact that she will always be along for the ride, pushing us both to inconceivable heights.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Irrigated agriculture comprises 13% (22.5 million hectares) of all arable land in the United States, yet irrigated crop production accounts for approximately 49% of the market value of all crops in the U.S. (Hoffman & Evans, 2007) and plays an important role in the production of these staples. In semi-arid areas much like the western U.S., irrigation increases crop productivity by supplementing soil moisture levels between rainfall events, ultimately reducing crop stress. However, freshwater is a highly sought resource, placing large competition for beneficial water rights under the prior appropriations doctrine. Increasing urban sprawl, state compact requirements (Colorado Water Conservation Board), and unforeseen drought circumstances (Wines, 2014) all place additional weight on top of an already unstable water supply. Agriculture consumes approximately 80% of freshwater available, ranking as the prime user in the U.S. (Hoffman & Evans, 2007). Agriculturalists must adopt practices of effective water allocations, placing the correct amount of water in the soil profile at the right time. Although technology may not be the sole answer to this challenge, it will certainly play a key role in the solution of ensuring optimum crop productivity with limited water supplies.

The need to replenish soil moisture is derived from plant water usage and soil moisture evaporation. Together these two terms are usually combined into evapotranspiration (ET). The large role of ET within an agricultural system is precisely the reason why scientists attempt to calculate ET values. Numerous models have tried to predict this value, the most recent of which encompass solar radiation, wind speed, temperature, relative humidity, latitude, longitude, day of the year, and elevation within the equation. As outlined in the American Society of Civil

Engineers (ASCE) manual, the 2005 standardized ASCE Penman-Monteith equation is currently the typical equation used within agriculture today (ASCE-EWRI, 2005). To allow broader geographic representation and usage of this equation, two reference ET values are given, both of well watered crops without disease or stress: 50cm tall, full canopy alfalfa (ET_r) and 12cm tall, cool season clipped grass (ET_o) (Allen et al., 2007). These two reference crops allow standardized crop dynamics, such as canopy conductance, which in turn can lead to simplified calculations from common weather parameters. The alfalfa reference value is assumed analogous to most agricultural crops at full canopy, or maximum ET rates (Pereira et al., 1999). To convert these reference values into useful data for a specific crop in a certain region, a crop coefficient (K_{cr} for alfalfa-based ET or K_{co} for grass-based ET) is multiplied with ET_r to calculate crop evapotranspiration (ET_c). The K_c values vary throughout the growing season and can range from near 0 to slightly above 1 in the case of a grass reference (usually maximum of 1 for alfalfa reference). The magnitude of K_c varies throughout the growing season and is dependent on vegetative leaf area (Doorenbos and Pruitt, 1977). Since solar radiation is the main driver of the vaporization of water molecules (latent heat flux) within the plant stomata, the leaf area index (the amount of leaf surface area per land area) determines the magnitude of the crop coefficient. Actual evapotranspiration (ET_{c_actual}) is calculated by multiplying an additional crop stress coefficient (K_s) to the crop ET equation (Eq. 1.1). A crop stress variable ranges from 0 to 1, with 1 representing no plant stress (Allen et al., 1998).

$$ET_{c\_actual} = ET_r \times K_{cr} \times K_s \qquad \text{Eq. (1.1)}$$

where ET_{c_actual} is the actual crop ET, ET_r is the reference ET value, K_{cr} is the alfalfa-based crop coefficient, and K_s is the stress coefficient.

The soil water balance uses actual crop ET values as well as soil water deficit, precipitation and net irrigation amounts to predict the soil water deficit or the depletion of soil water from field capacity on a daily basis (Allen et al., 1998). The law of conservation of mass states that no matter can be created or destroyed only reallocated to different physical states or locations. With this law in mind, moisture can only exit the soil profile system via ET or deep percolation and can only enter the system through precipitation, irrigation, or up flux from shallow ground water (Andales et al., 2011). This approach enables the prediction of soil water depletion on a daily basis, which is the net irrigation requirement that can be used in irrigation scheduling.

WISE (Water Irrigation Scheduling for Efficient Application; Andales et al., 2014), a model that uses the soil water balance theory, has been created on a cloud based platform called eRAMS (Environmental Risk and Analysis Management System; Arabi, 2011; www.erams.com). Similar computer programs have been created such as: KanSched2 (http://www.mobileirrigationlab.com/kansched2), NDAWN (http://ndawn.ndsu.nodak.edu/irrsched-crop-view-form.html), and the Belle Fourche Irrigation District scheduling tool (Oswald & Werner, 2009), however, WISE has the ability to incorporate different weather networks to enable expansion of different geographic regions with various crop types. Since WISE utilizes the soil water balance, which in turn requires ET_{c_actual} values, there is a need for validation of crop coefficients in this spatial region to accurately estimate ET_c.

The purpose of this project was to improve the accuracy and functionality of WISE for irrigation scheduling. The first objective was to modify an existing K_{cr} curve for sugar beets for use in northeastern Colorado. Modifying this curve will improve the accuracy of calculated sugar beet ET_c and in turn will increase precision of irrigation scheduling for sugar beets within WISE.

Chapter 2 is a manuscript on the sugar beet K_{cr} study conducted during the growing seasons of 2013 and 2014 in northeastern Colorado. The manuscript was formatted for future submission to *Agricultural Water Management*. Crop coefficient values modified for northeast Colorado conditions have been incorporated into WISE for future use. For wide adoption of any software program, accuracy is one of the greatest contributing factors; however, convenience of use is also highly significant. The second objective was to include basic irrigation scheduling functions of WISE into a mobile application. iPhone and iPad applications were created, giving users the ability to access basic irrigation scheduling information from any location with mobile data connectivity. Another manuscript describing the development and functionality of this app is presented in the third chapter. The manuscript was also formatted for future submission to *Computers and Electronics in Agriculture* as an application note which limited the length to eight double spaced pages.

REFERENCES

Allen, R.G., Pereira, L.S., Raes, D., Smith, M., 1998. Crop evapotranspiration – Guidelines for computing crop water requirements – *FAO Irrigation and drainage paper 56*. Food and Agriculture Organization of the United Nations, Rome, Italy: FAO.

Allen, R.G., Wright, J.L., Pruitt, W.O., Pereira, L.S., Jensen, M.E., 2007. Water Requirements. In: *Design and Operation of Farm Irrigation Systems. 2nd ed*. St. Joseph, MI. American Society of Agricultural and Biological Engineers.

Andales, A.A., Bauder, T.A., Arabi, M. 2014. A Mobile Irrigation Water Management System Using a Collaborative GIS and Weather Station Networks. In: Ahuja, L.R., Ma, L., Lascano, R., *Applications of Agricultural System Models to Optimize the Use of Limited Water, Volume 6*. (In Press).

Andales, A.A. Chávez, J.L., Bauder T.A., 2011. Irrigation Scheduling: The Water Balance Approach. *Colorado State University Extension Service Fact Sheet no. 4.707*. Available online at: http://www.ext.colostate.edu/pubs/crops/04707.html. (verified 5 Nov. 2014).

Arabi, M. 2011. Environmental Risk Assessment and Management System. *Colorado Water, Vol. 28, Issue 1*, pp. 15-17. (Available online at http://wsnet.colostate.edu/cwis31/ColoradoWater/Images/Newsletters/2011/CW_28_1.pdf) (verified 5 Nov. 2014).

ASCE-EWRI. 2005. The ASCE Standardized Reference Evapotranspiration Equation. Report 0-7844-0805-X, *ASCE Task Committee on Standardization of Reference Evapotranspiration*. Reston, Va.: American Soc. Civil Engineers.

Colorado Water Conservation Board. (n.d.). *Interstate Water Allocation*. Retrieved 03 28, 2014, from Department of Natural Resources: http://cwcb.state.co.us/legal/Documents/InterstateCompacts/CompactFacts.pdf. (verified 5 Nov. 2014).

Doorenbos, J., Pruitt, W.O., 1977. Guidelines for prediction of crop water requirements. *FAO Irrigation and Drain. Paper No. 24 (revised)*, Rome, Italy: United Nations FAO.

Hoffman, G.J., Evans, R.G., 2007. Introduction. In: *Design and Operation of Farm Irrigation Systems. 2nd ed*. St. Joseph, MI. American Society of Agricultural and Biological Engineers.

Oswald, J. and Werner, H., 2009. On-Line Irrigation Scheduling within the Belle Fourche Irrigation District. *World Environmental and Water Resources Congress 2009*: pp. 1-10.

Pereira, L., Perrier, A., Allen, R., and Alves, I. (1999). ”Evapotranspiration: Concepts and Future Trends.” *J. Irrig. Drain Eng.*, 125(2), 45–51.

Wines, Michael. “West’s Drought and Growth Intensify Conflict Over Water Rights.” *The New York Times* [New York City] 17 Mar. 2014: A1. Print.

CHAPTER 2: MODIFYING A SUGAR BEET CROP COEFFICIENT CURVE FOR A SEMI-ARID CLIMATE IN NORTHEASTERN COLORADO

SYNOPSIS

Crop coefficient curves permit the conversion of a standardized reference evapotranspiration (ET_r) rate to actual crop ET (ET_c), a value critical for irrigation planning and management. The objective of this study was to modify an existing alfalfa reference-based sugar beet crop coefficient (K_{cr}) curve suitable for the semi-arid climate of Northeast Colorado. Sugar beet (*Beta vulgaris*) crop progress, leaf area index (LAI), root zone soil moisture, and growing degree days (GDD) were measured throughout the 2013-14 growing season in four locations across Northeast Colorado. Using a soil water balance, the modeled daily soil deficits (D) of the root zone using both the original and modified K_{cr} curve were compared to observed D and the K_{cr} curve inflections were adjusted according to the measured sugar beet soil water depletions. Full canopy was achieved later than originally predicted, causing an increase in modeled ET rates over actual ET_c and requiring the time of attaining peak K_{cr} (cutoff 2) to be adjusted from 33% to 43% of maturity (represented as total GDD for maturity). Actual ET_c was reduced earlier than modeled using the original K_{cr} curve inducing the need to adjust the beginning of the senescence stage (cutoff 3) from 83% to 69% of maturity. Both changes shortened the length of the full canopy phase, thus significantly decreasing calculated ET_c. Upon modifications of the curve, relative error (RE) of the predicted soil water deficits over all sites from the original to the modified crop coefficient were reduced from RE values ranging from 11% - 300% down to RE values ranging from 0% - 265%. Likewise, the mean bias error over all fields was reduced from -23.73 mm for the original curve to -15.91 mm using the modified curve. Large errors were

caused by uncertainties in soil properties, effects of hail damage on actual leaf area and ET_c, spatial variability in precipitation or irrigation, and differences in field micro-climate and measured weather station data. The suggested modifications to the crop coefficient curve will provide increased accuracy in the calculation of sugar beet ET_c for the northeast Colorado region.

INTRODUCTION

Viable crops grown in semi-arid climates usually require irrigation between precipitation events to increase soil moisture ultimately to prevent crop stress which in turn improves crop vigor and yield. A proper irrigation schedule can increase yield and profit while decreasing the possibility of wasted water, energy and labor (Broner, 2005). Sugar beets are a prime commodity in the United States, raised for the extraction of sucrose. The majority of sugar beet acres in the western US require irrigation applications to produce ample yields (USDA Staff, 2012). Approximately 55% of all sucrose production in the United States is comprised from sugar beets, thus, improving sugar beet irrigation schedules could increase sugar production within the U.S. while saving water amounts and quality (USDA Staff, 2012).

The American Society of Civil Engineers—Environmental and Water Resources Institute (2005) developed a standardized method to calculate alfalfa reference evapotranspiration (ET_r). This reference provides a baseline for crop water usage. As outlined by Allen et al., (1998), in the Food and Agriculture Organization of the United Nations, paper 56 on guidelines for computing crop water requirements (FAO-56), applying a crop coefficient (K_c) as a factor of ET_r, allows for the calculation of ET_c . Equation 2.1 exhibits the connection between reference and crop ET:

$$ET_c = ET_r \times K_{cr} \times K_s \quad \text{Eq. (2.1)}$$

where ET_c is the crop ET (mm), ET_r is the alfalfa reference ET (mm), K_{cr} is the alfalfa based crop coefficient for the specific crop, and K_s is the stress coefficient that ranges from 0 for extreme stress to 1 for no stress. A pre-set management allowed depletion (MAD) percentage for the root zone determines the most optimal time period for irrigation in order to prevent plant stress (Merriam, 1966). A MAD of 50% for sugar beets allows the moisture of the root zone to be half depleted before the plant encounters stress. This is calculated by subtracting the wilting point (WP; soil tension where plants can no longer uptake water) from field capacity (FC; water is held by soil pores and solids, no gravitational flow) and multiplying by MAD ([F.C. – W.P.] x MAD). The sugar beet MAD was adjusted from the FAO-56 suggested 55% to reduce the probability of plant stress (Allen et al., 1998). The sugar beet root zone was assumed to have a moderate value of 91.4 cm (Allen et al., 1998). As the soil water deficit falls below the root zone allowed depletion (dMAD), a linear stress function is applied as the K_s term (figure 2.1; $0 < K_s < 1$).

This study utilized an existing ET-based irrigation scheduler that calculates a daily soil water balance (Gleason, 2013). The water balance is a simple concept of accounting where inputs are added and outputs are subtracted, yielding an absolute value of soil water depletion of the root zone. The calculation is stated in FAO Irrigation and Drainage Paper number 56 equation (85) and is modified as (Allen et al., 1998):

$$D_c = D_P + ET_c - P - Irr - U + SRO + DP \quad \text{Eq. (2.2)}$$

where: D_c is the soil water deficit at the end of the current day, D_p is soil water deficit from the previous day, ET_c is actual crop evapotranspiration for the current day, P is gross precipitation

for the current day, Irr is the net amount irrigated on the current day, U is the current days upflux, SRO is surface runoff for the current day, and DP is deep percolation on the current day where all units are expressed as a depth of water (mm). An initial measured D_c of the root zone enables increased accuracy with this accounting equation. During precipitation events, a curve number is applied to estimate the SRO component (NRCS, 2004). Within this equation, P can be easily recorded using rain gauges and the irrigation manager can track gross irrigation. Up flux, SRO, and DP are difficult to measure, therefore, according to Andales et al. (2011), it is possible to account for these variables in a simplified way as:

$$D_c = D_P + ET_c - P - Irr \qquad \text{Eq. (2.3)}$$

and assume that when D_c is negative, the absolute value of that number is an estimate of DP + SRO. D_c is predicted as zero when (P + Irr) is greater than (D_P + ET_c). Upflux (U) from a shallow water table is neglected. This provides an easy accounting formula for most irrigated fields that are not influenced by a shallow water table, without having knowledge of P or Irr intensity.

The scheduling program used to compute this equation was originally based in Microsoft Excel® (Colorado Irrigation Scheduler – Annuals or CIS-A) but has since expanded to a cloud-based platform (Gleason, 2013; Andales et al., 2014). Soil data including FC and available water content (AWC) are retrieved via a query from the United States Department of Agriculture (USDA) Natural Resource Conservation Service (NRSC) Web Soil Survey (Soil Survey Staff, 2013). Nearby Colorado Agriculture Meteorological Network (CoAgMet; http://www.coagmet.colostate.edu) and Northern Colorado Water Conservancy District (Northern Water or NCWCD; http://www.northernwater.org/WaterConservation/WeatherandETInfo.aspx) weather station data

is downloaded and updated daily. The user has the ability to add crop attributes such as crop type, planting, emergence, and harvest dates, rooting depth, irrigation efficiency, growing degree-days at maturity, soil water measurement corrections, and elements of the crop coefficient curve. Throughout the season, the user is required to insert gross irrigation and unrecorded or corrected rainfall data to maintain the accuracy of the water balance calculations.

K_{cr} curves are separated temporally into four different phases using three cutoff values (Doorenbos & Pruitt, 1977). The cutoffs in this paper will be listed as percent of growing degree days (GDD) of plant maturity. The magnitude of the K_{cr} curve is controlled by the K_c initial, mid, and end values (figure 2.2). Sugar beet K_{cr} curves can be found in other texts such as Allen et al., (2007), and Wright J. L. (1982); however, these values are presented temporally as percent time from planting to effective full cover and days after effective full cover date, neither of which is the CIS-A required percent maturity. There are no known values for the specific region of Northeast Colorado. An original sugar beet K_{co} (referred to as original curve) empirically derived from grass reference (ET_o) values in FAO-56, was adjusted for an alfalfa reference by dividing all K_c values by a moderate number of 1.2 to account for change in canopy conductance and crop height (suggested conversion values of 1.1 for humid and 1.4 for arid areas; Allen et al., 1998). This original curve was used within the soil water balance model and tested against observed soil moisture values.

The objective of this study was to modify a sugar beet, alfalfa based reference, K_{cr} curve for the semi-arid climate of Northeast Colorado. The original sugar beet K_{cr} curve was tested by comparing measured soil deficits against the model predicted values. Modifying this K_{cr} curve for sugar beets to represent growing conditions and climate in Northeast Colorado will increase

the accuracy of actual ET_c calculations and ultimately lead to more precise estimates of irrigation water requirements.

MATERIALS AND METHODS

Research Locations

Four geographic locations were selected in Northeast Colorado during the 2013 and 2014 growing season. Exact fields were not repeated between years in order to reduce disease persistence; however, all fields were at least within 2 km of the previous year. All crop fields were planted to sugar beets between April 27 and May 13, 2013, during the initial year and between April 10 and April 23, 2014, for the second. Tillage management varied from each location but all fields were intended to be fully irrigated with 76.2 cm (30 inch) wide rows. The fields were in close proximity to the Colorado towns of: Gilcrest (latitude N40.28601°, longitude W104.79503°, elevation 1435.7m above mean sea level), Hillrose (N40.34707°, W103.50022°, 1245.01m), Vernon (N39.98772°, W102.28608°, 1162.86m), and Wellington (N40.7513°, W105.02209°, 1606.29m) (Figure 2.3). For redundancy, two sampling points for soil moisture and plant monitoring were selected inside each field, located within the same row 15.2 m apart.

Irrigation Management

Center pivot sprinkler irrigation was utilized at all locations. The irrigation application efficiencies of these systems were assumed to be established at 90%, according to the Colorado High Plains Irrigation Practices Guide (Rachel Barta, 2004). Two, 1-liter catch cans were placed by each sampling point (a total of four catch cans per field) to measure weekly gross irrigation applications. In 2013 the Hillrose location and in 2014 the Vernon field utilized a mid-elevation spray application (MESA) sprinkler, where the low height placement of the nozzles prevented

catch cans from accurately recording gross irrigation amounts. Irrigation logs were maintained, recording both timing and amount. Precipitation values were recorded immediately outside the field boundaries with an Onset RG3 tipping bucket rain gauge (Onset Computer Corporation, Bourne, MA), logging the time for every 0.254 mm (0.01 inch) of rainfall and summed over a 24 hour period to produce daily precipitation. During most of the growing season, fields were fully irrigated and without disease; however, on occasion due to water shortage, the irrigation managers allowed the plants to experience a slight degree of water stress.

Leaf Area Index

Leaf area index (LAI) was measured weekly using a LI-COR LAI 2000 meter (LI-COR, Inc., Lincoln, NE). This non-destructive method detects incoming light above and beneath the plant canopy using a 360° view and multiple zenith angles, correlating light interception to leaf area (Grantz et al., 1993). Three locations were selected in each field for the LAI measurement: near each sampling point and half-way between the two sampling points. Each measurement recorded and averaged four readings which were measured next to the row, 25%, 50% and 75% of the distance between rows. Full sunlight was often unavoidable and a 90° cap was placed on the lens to decrease the field of view to 270º and additional shade was provided via an umbrella, ultimately lowering the error by reducing the amount of direct sunlight contact with the lens (J.M. Welles, 1991). In accordance to Al-Kaisi and Broner (2009), full canopy was assumed at a LAI of 3 or greater.

Weekly nadir photos visually captured canopy cover. The images focused on a 76.2 cm PVC square frame to represent width between rows (30 inch row spacing), allowing a visual reference of full canopy cover. Broad field photos were captured as well.

Soil Properties

Soil bulk density measurements were taken in close proximity to each site just after plant emergence at 15, 45, and 75, cm depths using a Madera probe (Precision Machine Company Inc., Lincoln, NE). A total of 6 cores, at a volume of 61.7 cm^3 per core, were taken at each field, 3 for each sampling point. Bulk density measurements were calculated by the methods described in Grossman and Reinsch (2002), using the wet soil core samples. Soil texture cores were extracted and separated by visual changes within the profile. Soil texture was determined using a particle size analysis conducted with the hydrometer method (Gee and Or, 2002), and texture classifications were taken from the USDA NRCS soil texture triangle calculator (http://www.nrcs.usda.gov/wps/portal/nrcs/detail/national/soils/?cid=nrcs142p2_054167).

Available water content values were extracted from the NRCS SSURGO database (http://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2_053627) for each location and depth. Field capacity values were obtained subsequently from reviewing volumetric water content values calculated from the (bi-)weekly gravimetric samples. Soil moisture 24-48 hours after a large irrigation and/or precipitation event were used as FC measurements. The difference of the AWC from FC was used to estimate the WP value.

Soil Water Content Measurements

The initial soil volumetric water contents were taken from the bulk density cores (see previous section) as the initial root zone water content. Gravimetric sampling was conducted bi-weekly in 2013 and weekly in 2014 using a JMC Backsaver probe with a core diameter of 1.905cm (Clements Associates Inc., Newton, IA), at depths of 90 cm sampling in 15 cm intervals. The moist samples were weighed, dried, and re-weighed according to the gravimetric water content method in Topp and Ferré (2002). Volumetric water content values (cm^3 cm^{-3})

were obtained for each depth by taking the product of the gravimetric sample and the bulk density, divided by the density of water (1.0 g cm^{-3}).

Watermark electrical resistance sensors (Irrometer, Riverside, CA), were placed in both sites at each location at depths of 15, 45, and 75 cm. Hansen data loggers (M.K. Hansen Co, Wenatchee, WA) recorded the soil tension on 8 hour intervals. This data was used as a guide and reference to confirm irrigation timing.

Meteorological Measurements

Hourly weather parameters used in calculating ET including wind speed, relative humidity, temperature, and solar radiation, were obtained by nearby CoAgMet and Northern Colorado Water Conservancy District weather stations. Table 2.1 displays the distance from each field site to the closest weather station(s). ET_r was computed using the American Society of Civil Engineers Environmental and Water Resources Institute (2005) standardized Penman-Monteith equation in REF-ET, a program created by the University of Idaho (http://extension.uidaho.edu/kimberly/2013/04/ref-et-reference-evapotranspiration-calculator/). The ASCE equation is:

$$ET_{ref} = \frac{0.408\Delta(R_n - G) + \gamma \frac{C_n}{T+273}\mu_2(e_s - e_a)}{\Delta + \gamma(1 + C_d\mu_2)} \quad \text{Eq. (2.4)}$$

where: ET_{ref} is the standardized ET_r (mm/h), R_n is calculated net radiation at the crop surface (MJ m^{-2} h^{-1}), G is the soil heat flux density at the soil surface (MJ m^{-2} h^{-1}), T is the mean hourly air temperature (ºC), μ_2 is mean hourly wind speed (m s^{-1}), e_s is the saturation vapor pressure (kPa), e_a is mean actual vapor pressure (kPa), Δ is the slope of the saturation vapor pressure-temperature curve (kPa $ºC^{-1}$), γ is psychrometric constant (kPa $ºC^{-1}$), C_n is numerator constant that changes with reference type and calculation time step (K mm s^3 Mg^{-1} h^{-1}), C_d is denominator

constant that changes with reference type and calculation time step (s m^{-1}) and units for the 0.408 coefficient are (m^2 mm MJ^{-1}). Hourly values were summed over a 24-hour period to produce daily reference ET. For both years at the Hillrose location, two weather stations were in close proximity of the field and an inverse distance weighted average of the ET values was calculated using the formula as defined by Shepard (1968). Equation 2.5 has been adapted for this project:

$$ET_{ridw} = \frac{\sum_{i=1}^{N} \frac{M}{d^2}}{\sum_{i=1}^{N} \frac{1}{d^2}} \quad \text{Eq. (2.5)}$$

where: ET_{ridw} is alfalfa reference inverse distance weighted average ET (mm), M is the measured ET_r of each location (mm), and d is the distance from the weather station to the crop field (m). To obtain ET_c (mm), ET_r is multiplied by the crop coefficient and stress coefficient as shown above in equation 2.1.

A model E atmometer (ETgage Company, Loveland, CO) equipped with a canvas #54 (alfalfa) cover was placed outside each field location, 1 meter above vegetation to provide on-site ET_r comparisons. Errors induced by high temperatures and low humidity were remediated by using an adjustment equation:

$$ET_{gage\ adj} = 2.868 + 1.006 \times ET_{gage} - 0.06 \times T_{max} \quad \text{Eq. (2.6)}$$

where: $ET_{gage\ adj}$ is the adjusted daily value of ET_r (mm day^{-1}), ETgage is the daily ET rate from the atmometer (mm day^{-1}), and T_{max} is the maximum temperature (ºC) (Gleason et al., 2013). Comparisons of total ET_r values from atmometers and weather stations are shown in table 2.2. Atmometer data loggers contained a temperature thermistor and were placed inside a solar radiation shield to record ambient air temperatures.

Growing degree-days commence and accumulate following plant emergence using the equation:

$$GDD = \frac{T_{max}+T_{min}}{2} - T_{base} \quad \text{Eq. (2.7)}$$

where: T_{max} is the maximum measured temperature (ºC), T_{min} is the recorded minimum temperature (ºC), and T_{base} is the pre-set base temperature (ºC). Neither the high nor the low temperature can fall above or below the pre-set maximum temperature of 30.0 (ºC) and the base temperature of 1.1 (ºC) (Holen and Dexter, 1997). GDD at maturity was derived by taking previous temperature data and typical planting and harvest dates and solving total GDD. For irrigation scheduling, a total of 2944.4 growing degree-days were deemed as full maturity for the initial year of sugar beets. On-site ambient air temperature measurements were used for calculating GDD. These values were compared to GDD calculated from nearby weather station values. Table 2.3 compares daily GDD coefficient of determination (r^2), total GDD units, and ending maturity.

Colorado Irrigation Scheduler – Annuals

Colorado Irrigation Scheduler – Annuals was used in this study to output predicted soil water deficits resulting from the use of both the original and modified crop coefficient curves. The predicted soil water deficits were compared to observed soil deficit values calculated from observed volumetric moisture for the entire root zone depth. Soil properties, calculated ET_r, gross irrigations, and precipitation were all inputted as model parameters into the spreadsheet. Original deficit values were recorded from the initial gravimetric samples and inputted into CIS-A. Total soil moisture in the root zone was calculated from the (bi-)weekly gravimetric values. The calculation for the deficit of the root zone is:

$$D = \left(\theta_{FC\_V} - \theta_V\right) * D_{rz} \quad \text{Eq. (2.8)}$$

where D is the deficit of the root zone (mm), θ_{FC_V} is the volumetric water content at field capacity ($cm^3\ cm^{-3}$), θ_V is the current volumetric water content ($cm^3\ cm^{-3}$), and D_{rz} is the depth of the root zone (mm). This value was compared to predicted values outputted by the CIS-A for both the original and modified crop coefficient curves.

Evaluation of the K_{cr} Curve

Observed soil water deficits for the root zone were compared against modeled deficits using the modified K_{cr} curve. Accuracy of the modeled soil water deficits were evaluated using the following statistics: mean absolute error (MAE), mean bias error (MBE), root mean squared error (RMSE), and relative error (RE), with the equations as follows:

$$MAE = N^{-1} \sum_{i=1}^{N} |P_i - O_i| \quad \text{Eq. (2.9)}$$

$$MBE = N^{-1} \sum_{i=1}^{N} (P_i - O_i) \quad \text{Eq. (2.10)}$$

$$RMSE = [N^{-1} \sum_{i=1}^{N} (P_i - O_i)^2]^{0.5} \quad \text{Eq. (2.11)}$$

$$RE\ \% = \frac{(\bar{P} - \bar{O})}{\bar{O}} \times 100 \quad \text{Eq. (2.12)}$$

where N is the number of observations, P is the predicted deficit, O is the observed deficit, and $\bar{P}$ and $\bar{O}$ are the mean predicted and observed deficit respectively. Both modified and original K_{cr} values are reviewed.

RESULTS AND DISCUSSION

In order to validate weather station measurements, ET_c for the Hillrose site (2013) was tested against multiple methods. Mean and standard deviation values were calculated from the Brush and Sterling NCWCD weather stations, adjusted atmometer readings, and Remote Sensing of ET (ReSET; Elhaddad et al., 2011) model to provide high and low ET_c readings (Elhaddad, 2014; figure 2.4). It was found that the regional weather stations were accurately measuring ET_r and GDD.

Predicted soil deficits derived using the original K_{cr} curve were compared against observed deficits. Sugar beets with a preset root zone, demonstrated that the predicted values overestimated the soil deficit by 76.01% over all the fields, the largest error being Hillrose East (2014) with a 299.52% overestimation. The most accurate site was Wellington South (2014) with a relative error of 11.07%. The average error over all fields (MBE) was -23.73 mm, (MAE) 28.79 mm, and (RMSE) 37.26 mm. Since negative deficit values were used within equations 2.9 thru 2.12, then a negative MBE or RE means that predicted D values were larger in magnitude (absolute value) compared to observed D values.

After reviewing the errors in predicted soil deficit (using the original K_{cr} curve), and compiling the data of percent maturity until full canopy, it was found that full vegetative development occurred later than predicted. This created an increase in modeled soil deficit since the second cutoff was premature in timing, thus the cutoff value was adjusted from 33% to 43% maturity, an average value when all fields reached full canopy with an LAI meter reading greater than 3 combined with photographic support. Errors between predicted and observed soil deficit values using the original coefficient curve also increased toward the end of the year as the modeled values overestimated actual water usage. Cutoff three was originally at 83% maturity,

over extending the length of the K_{c_mid} values. This cutoff was adjusted and shortened to 69% maturity, a value at which the lowest error occurred. Original and modified values are given in table 2.4.

Analysis of the modified K_{cr} against the same statistical tests showed that the predicted values overestimated the soil deficit by 50.95% when combining all fields during both growing seasons (table 2.5), which was reduced from 76.01% of the original curve. The largest over and underestimation was Hillrose East (2013) site with a RE of 265.15%, and Gilcrest North (2013) at -6.08% respectively. The most minimal relative error was Wellington South (2014) with an over prediction of 0.09%. Average error (MBE) showed an underestimation of 15.91 mm for a combination of all sites. MAE was 26.45 mm while RMSE was 34.94 mm. Figures 2.5 and 2.6 show predicted deficits using the modified curve, are plotted against observed values in two field locations: Gilcrest South 2013 (minimal relative error), and Hillrose East 2014 (greatest relative error). Figure 2.7 displays D_c from WISE using the modified crop coefficient superimposed over WaterMark tension data from the Hillrose North (2013) site. The deficit is consistent with soil tension values suggesting the tool and K_{cr} are working properly. Table 2.6 shows the water balance values for the Gilcrest South location 6/13/13 until 9/2/13 and Gilcrest North 6/4/14 until 9/4/14. Between September 10, 2013, and September 16, 2013, the Gilcrest field received 129 mm (5.08 inches) of rain, creating large deep percolation and surface runoff values which gave sound reason to leave these dates out of the water balance summary. No irrigations were applied after these dates. The change in storage (ΔS) was calculated from the other components of table 2.6. The actual change in storage in 2013 was measured by the volumetric water content and was 30.48mm.

There are several possible explanations for the large error of predicted soil deficit with

the modified K_{cr} curve. In 2013, Hillrose and Wellington, and once again in 2014 at Wellington, each field received at least one hail event during the season prior to 7/11/13, 8/5/13, and 8/4/14 respectively. Hail damage reduces the amount of leaf area available to intercept solar radiation, which drives plant transpiration, as well as slows plant growth, delaying actual growth stage results (Allen et al., 2007; Al-Kaisi & Broner, 2009). Reducing these factors slowed the rate of soil moisture depletion within the profile while the model prediction was unable to adjust the rate of ET_c to account for hail damage. Overestimations of the soil deficit were prevalent at all locations after the given dates. This can visually be seen in figure 2.8, which displays accumulated ET_c for Wellington 2013 after a hail event. The crop coefficient derived ET_c continues its normal calculations while the ReSET model accounts for leaf defoliation (Elhaddad, 2014).

Secondly, a more coarse gravel textured material greater than 2 mm was noticed on the north site of the Wellington field (in both 2013 and 2014), potentially affecting the bulk density and particle size analysis (PSA) results (Gee and Or, 2002). Since the PSA measurements were taken with soil that was sieved through a 2 mm screen, large gravel structures were not accounted for in this analysis leading to a misrepresentation of field capacity and available water content at this site.

Since the Hillrose field soil texture was a very fine mix between clay and a clay loam, field capacity values for the entire profile were difficult to determine (Smith and Warrick, 2007). In 2013 due to the high soil porosity and low conductivity, there was no noticeable irrigation or precipitation event that infiltrated the entire soil profile near the same date at this location. Soil gravimetric sampling at an increased frequency could have avoided this uncertainty. Also, irrigation application efficiencies were estimated to be 90%. An error within this term could

cause substantial error in predicted deficits.

As explained by Allen (1996), wind speed can be highly variable over large spatial distances, factoring in a possible error of the reference ET equation. Since the Vernon weather station (Idalia-CoAgMet) was 28.2 km away, wind measurement error could have been increased causing unknown differences in reference ET values. Beets at this site in 2013 were also replanted due to an early freeze, lowering the potential total growing degree days. Predicted values using the CIS-A tool were obtained by replacing the original planting date with the replanting date.

Precipitation data from regional weather stations was inaccurate for actual field conditions. On-site rainfall data replaced queried data within CIS-A. Table 2.7 presents the difference in total precipitation from these two different measurements. In 2013, a large rainfall event in early September was also separated due to the increase in errors.

Possible error could also be accounted for in the crop stress term. All field sites were intended to be fully irrigated; however, on occasion due to a lack of water resources, fields experienced stress. Since this study did not incorporate testing for the K_s term, it is unknown how large of an error was produced with this linear strategy. Plant stress is often unavoidable in agricultural production, thus, further research will be required to improve K_s when using the crop coefficient and soil water balance approach (Shrestha et al., 2010).

CONCLUSION

A soil water balance approach was used to validate ET_c calculations in four sugar beet fields across Northeast Colorado over two growing seasons. LAI, soil moisture, precipitation, and irrigation amounts were recorded. CIS-A was used in predicting daily soil deficits. Predicted

soil water deficits derived using an original sugar beet K_{cr} curve were tested against observed deficits and resulted in a 76.01% relative error. It was observed that the K_{cr} lead to over-predictions in ET_c during the full canopy and senescence growth stages. To modify the coefficient, the second and third cutoffs were adjusted from 33% and 83% to 43% and 69% maturity, respectively. Predicted deficits with this new K_{cr} curve were once again contrasted with the observed measurements. The modifications lowered the relative error range from 0% - 265% with a MBE over all sites of -15.91 mm.

Observed error between the actual and modeled soil deficits were partially accounted for and taken into consideration. Hail damage in three fields slowed ET_c rates while the predicted deficit curves maintained a higher rate. Course textured sands could have also led to inaccuracies in observed values at two of the locations.

It is obvious that mid-season soil moisture measurements used to correct predicted values will reduce error throughout the season, especially during or following a disease or canopy-damaging event. Further work needs to be conducted to verify the modified K_{cr} curve for sugar beets in 55.9 cm (22”) row spacing; however, this row spacing is less common in Colorado. It is recommended to increase the number of sampling points in each field to incorporate spatial soil, plant growth, and irrigation variability. Additionally, more detailed soils information for each site would account for spatial variability. This study evaluated an original K_{cr} curve and made successful modifications that resulted in a new sugar beet K_{cr} curve appropriate for conditions in Northeast Colorado. The modified sugar beet K_{cr} curve can be used with alfalfa reference ET_r calculated from the ASCE Standardized reference ET equation.

TABLES AND FIGURES

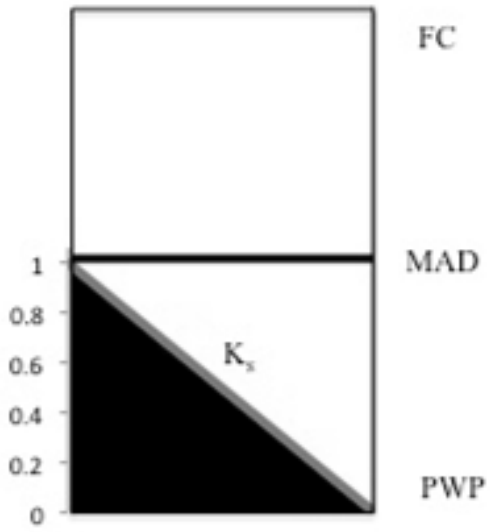


Figure 2.1 – A soil profile view, from field capacity to permanent wilting point. When the deficit of the root zone increases past the MAD, a linear stress coefficient (K_s) is applied to the ET_c calculation.

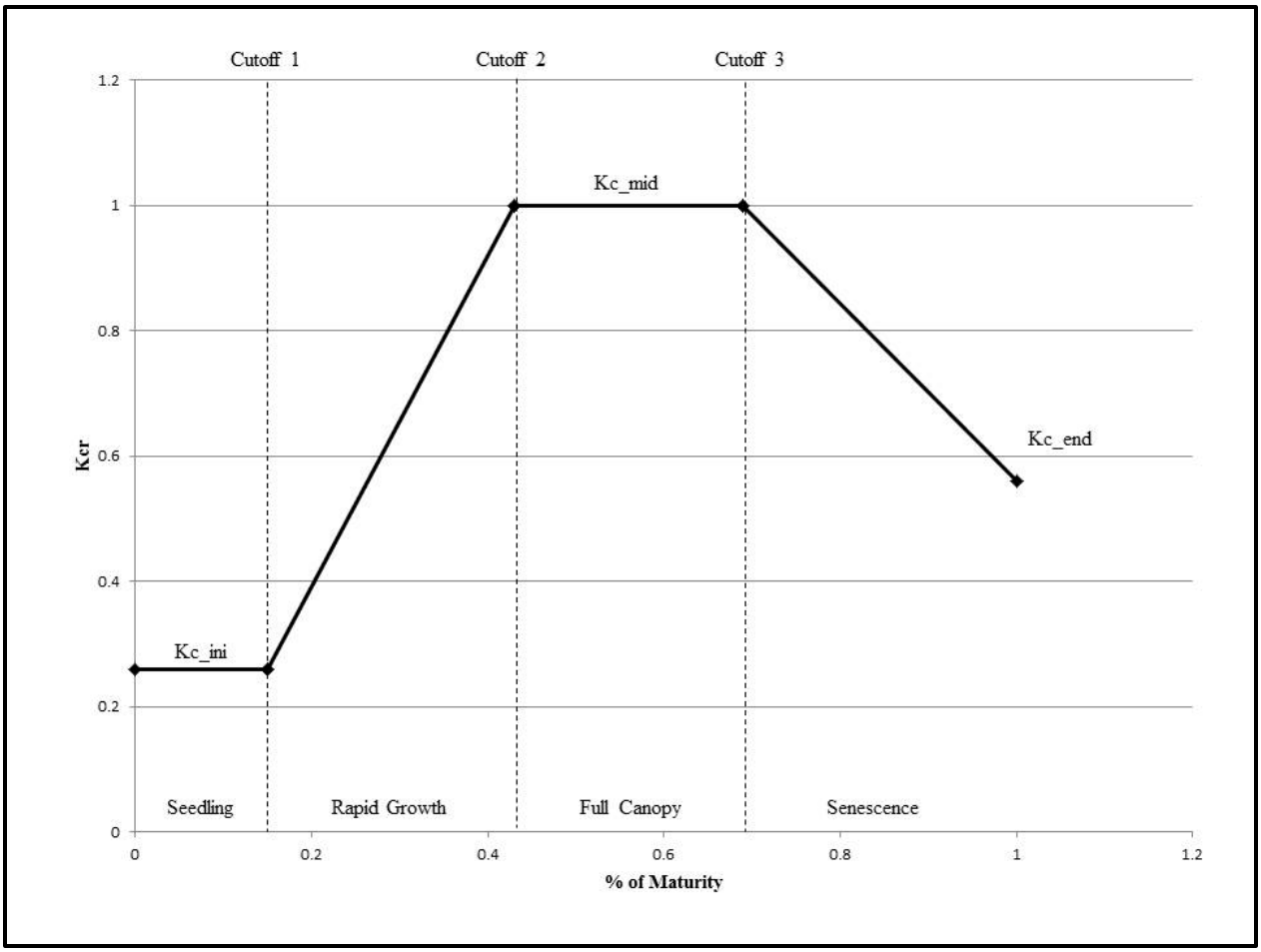


Figure 2.2 – The modified crop coefficient curve for sugar beets in Colorado using percent of maturity as the x-axis. Growth stages, K_c values, and cutoff dates are labeled.

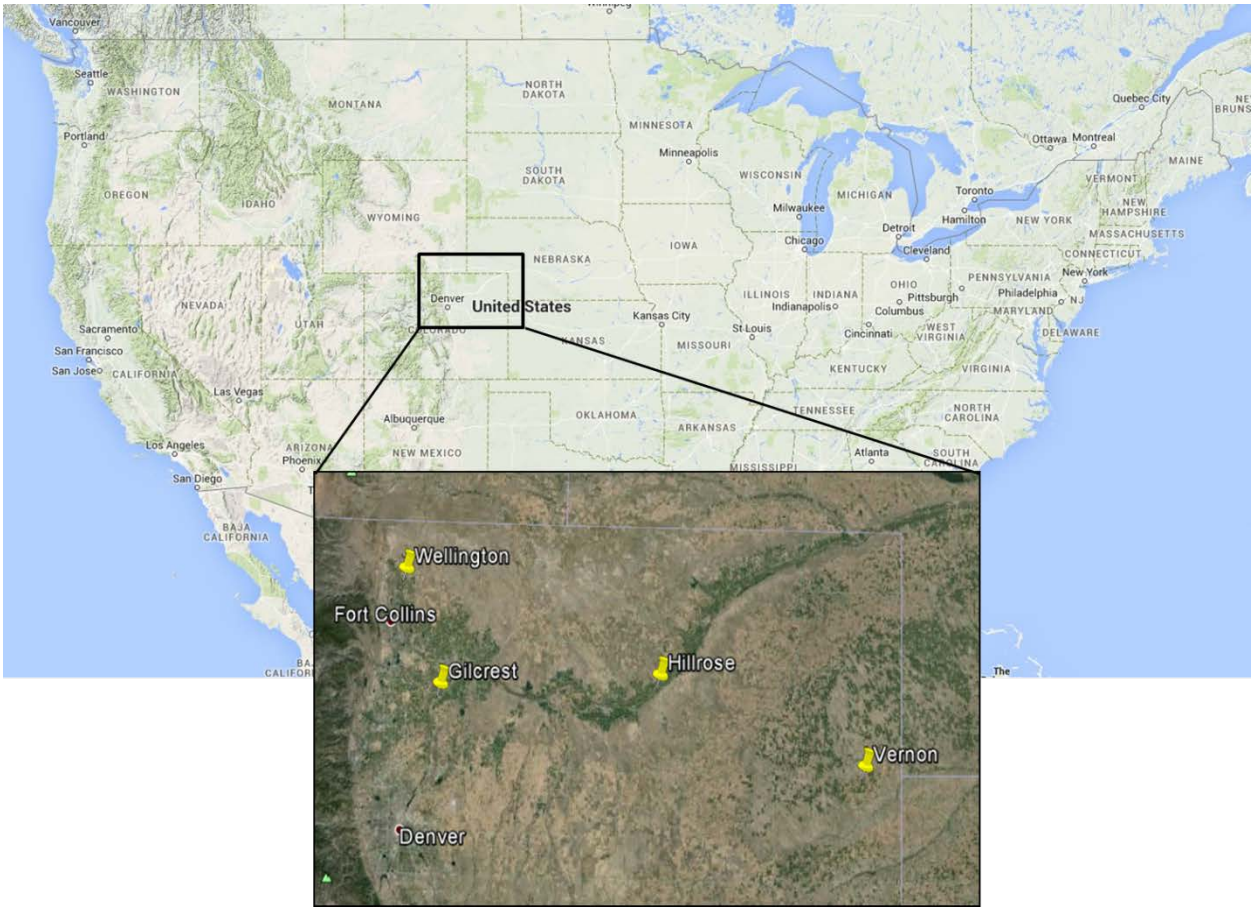


Figure 2.3 – A map view of the four sugar beet field locations in Northeastern Colorado.

Table 2.1 – Distance from field sites to NCWCD and CoAgMet weather stations.

	Weather Station	2013 Distance (km)	2014 Distance (km)
Gilcrest	Gilcrest	4.3	4.0
Hillrose	Brush	20.3	22.0
	Sterling	33.6	32.0
Vernon	Idalia	28.2	28.2
Wellington	CSU-ARDEC	10.8	12.2
	NCWCD		
	CoAgMet		

Table 2.2 -- Comparison of ET_r accumulation from weather stations vs. Atmometer.

Field-Site	N[a]	r^{2b}	Total Difference[c] (mm)
		2013	
Gilcrest	134	0.78	43.94
Hillrose	100	0.71	51.05
Vernon	104	0.86	12.19
Wellington	135	0.90	82.30
		2014	
Gilcrest	118	0.66	21.08
Hillrose	104	0.76	18.80
Vernon	104	0.73	113.28
Wellington	95	0.75	7.62
> Weather Stations			
< Weather Stations			

[a]N = number of days; [b]r^2 = coefficient of determination; [c]Total difference = total difference of ET_r

Table 2.3 – Comparison of GDD accumulation from weather stations vs. onsite temperature data.

Field-Site	N[a]	r^{2b}	ΔGDD[c]	Δ% Maturity[d]
		2013		
Gilcrest	110	0.94	12.72	0.24%
Hillrose	100	0.66	88.52	1.67%
Vernon	103	0.94	8.94	0.17%
Wellington	134	0.95	83.27	1.57%
		2014		
Gilcrest	118	0.91	41.72	0.79%
Hillrose	104	0.93	9.44	0.18%
Vernon	100	0.96	6.27	0.12%
Wellington	112	0.86	41.50	0.78%
> Weather Stations				
< Weather Stations				

[a]N = number of days; [b]r^2 = coefficient of determination; [c]ΔGDD = difference in growing degree days; [d]Δ% Maturity = difference in percent maturity.

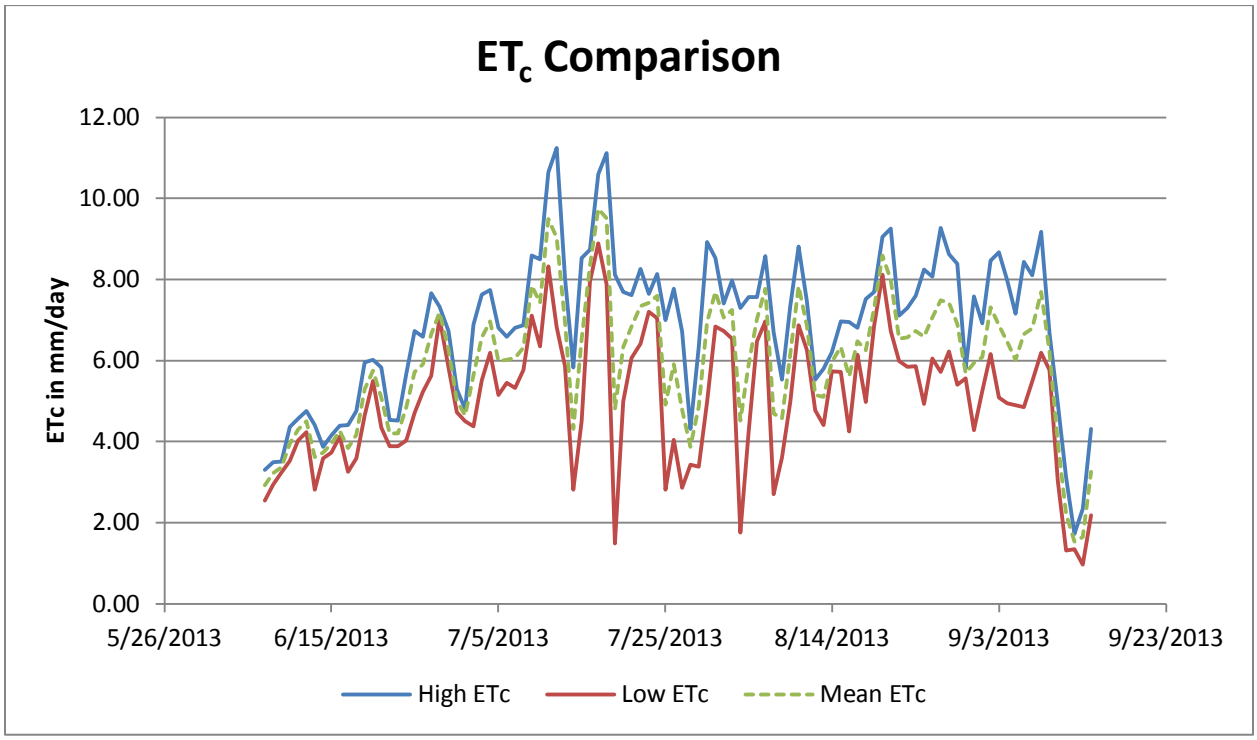


Figure 2.4 – A comparison of mean and ± 1 standard deviation (high and low) of ET_c at Hillrose in 2013. Values were calculated from two weather stations, atmometer (all three using K_{cr}), and the ReSET model.

Table 2.4 – K_{cr} table values for the original and modified curves.

Curve	% Maturity	K_c	K_c Value
Original			
Cutoff 1	15	Initial	0.26
Cutoff 2	33	Mid	1.00
Cutoff 3	83	End	0.56
Modified			
Cutoff 1	15	Initial	0.26
Cutoff 2	43	Mid	1.00
Cutoff 3	69	End	0.56

Table 2.5 – Comparisons of observed (from gravimetric method) and predicted soil moisture deficits for the 2013 and 2014 growing seasons.

Field-Site	N[a]	MAE[b] (mm)	MBE[c] (mm)	RMSE[d] (mm)	RE[e] (%)
		Original K_{cr} 2013			
Gilcrest North	7	14.68	-9.85	17.56	33.06
Gilcrest South	7	19.40	-13.71	22.60	49.12
Hillrose North	8	46.40	-42.04	51.39	174.01
Hillrose South	8	22.52	-8.80	26.21	15.34
Vernon East	9	30.78	-29.74	41.50	63.13
Vernon West	9	32.61	-15.83	38.01	27.28
Wellington North	9	42.99	-42.85	50.03	273.43
Wellington South	9	31.14	-29.74	38.78	96.61
All	**68**	**30.72**	**-24.77**	**38.32**	**67.54**
		Original K_{cr} 2014			
Gilcrest North	16	20.32	-17.90	25.74	98.54
Gilcrest South	16	24.83	-23.50	31.49	173.87
Hillrose East	14	31.53	-30.01	39.55	299.52
Hillrose West	14	29.63	-29.25	36.78	271.40
Vernon East	12	38.42	-27.12	45.79	40.72
Vernon West	12	61.76	-58.88	67.05	168.94
Wellington North	17	13.10	-7.87	17.88	20.57
Wellington South	17	15.67	-4.28	19.81	11.07
All	**118**	**27.71**	**-23.14**	**36.65**	**82.17**
2013 and 2014	**184**	**28.79**	**-23.73**	**37.26**	**76.01**
		Modified K_{cr} 2013			
Gilcrest North	7	11.45	1.81	14.85	-6.08
Gilcrest South	7	11.67	-5.09	16.40	18.23
Hillrose North	8	39.13	-34.77	43.42	143.92
Hillrose South	8	17.91	-1.54	20.29	2.67
Vernon East	9	28.19	-16.60	36.90	35.24
Vernon West	9	30.04	2.18	31.60	-3.76
Wellington North	9	35.86	-35.51	45.08	226.61
Wellington South	9	26.49	-22.73	37.53	73.84
All	**66**	**25.81**	**-14.66**	**33.49**	**39.96**
		Modified K_{cr} 2014			
Gilcrest North	16	18.68	-9.74	23.56	53.64
Gilcrest South	16	21.47	-15.54	29.53	115.02
Hillrose East	14	29.15	-26.56	38.26	265.15
Hillrose West	14	26.61	-25.81	34.57	239.44
Vernon East	12	45.99	-16.32	49.81	24.51

Vernon West	12	55.68	-48.08	62.50	137.96
Wellington North	17	14.23	-2.83	19.06	7.39
Wellington South	17	16.39	-0.04	21.38	0.09
All	**118**	**26.81**	**-16.60**	**35.73**	**58.96**
2013 and 2014	**184**	**26.45**	**-15.91**	**34.94**	**50.95**

[a]N = number of observations; [b]MAE = mean absolute error; [c]MBE = mean bias error; [d]RMSE = root mean square error; [e]RE = relative error.

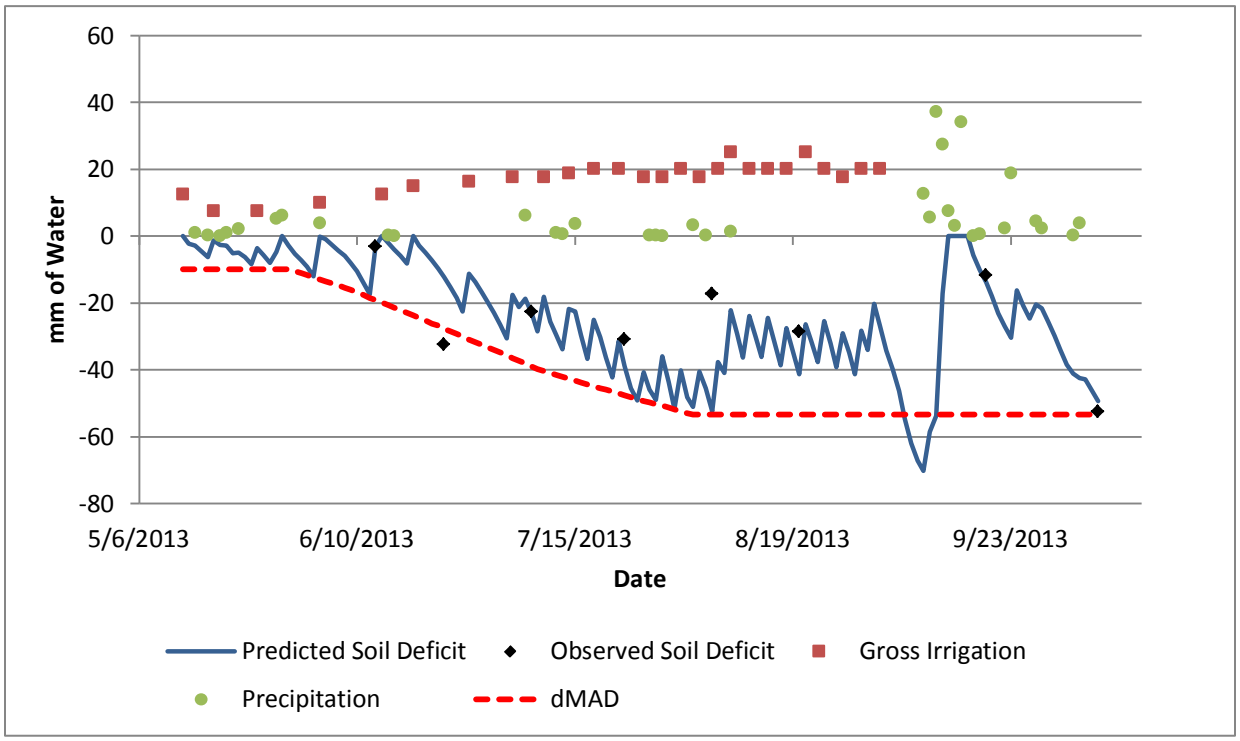


Figure 2.5 – Comparison of the modified crop coefficient predicted soil deficit (blue line) using the soil water balance equation against the observed values (black diamond's) for the Gilcrest South 2013 site. Also shown is the management allowed depletion of the root zone (dMAD; dashed red line), gross irrigation (red square's), and precipitation (green circle's). The managed root zone was 711 mm.

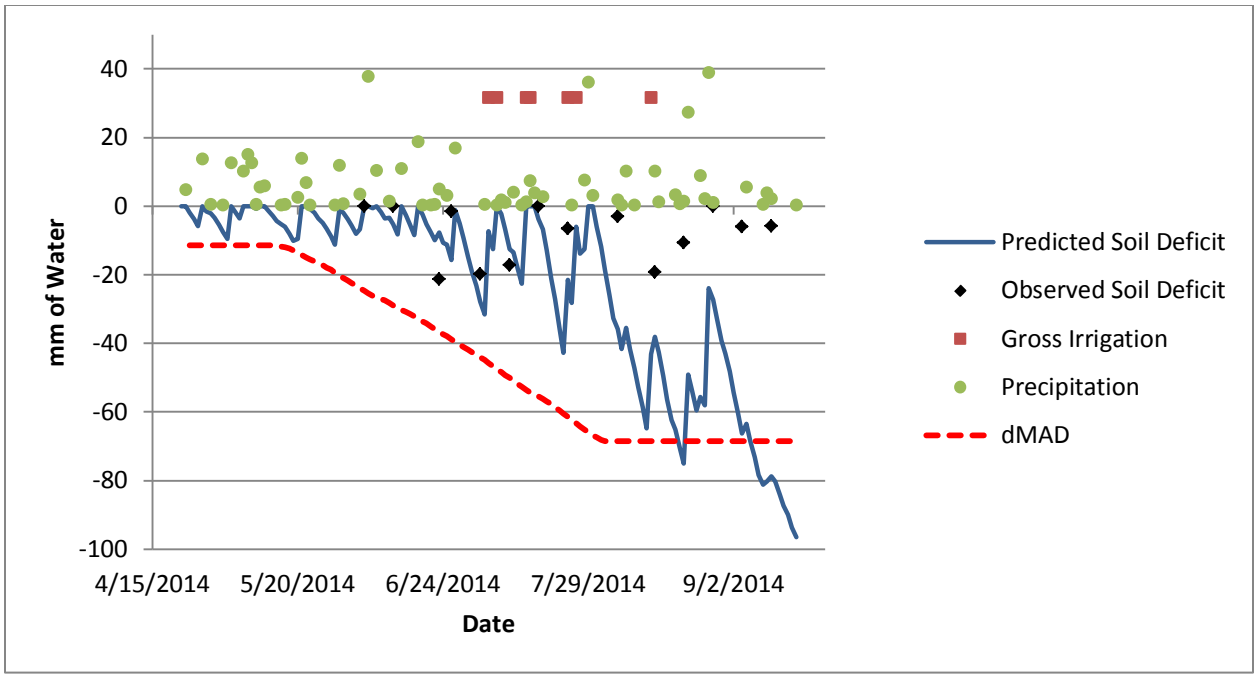


Figure 2.6 – Comparison of the modified crop coefficient predicted soil deficit (blue line) using the soil water balance equation against the observed values (black diamond's) for the Hillrose East 2014 site. Also shown is the management allowed depletion of the root zone (dMAD; dashed red line), gross irrigation (reds square), and precipitation (green circle's). The managed root zone was 711 mm.

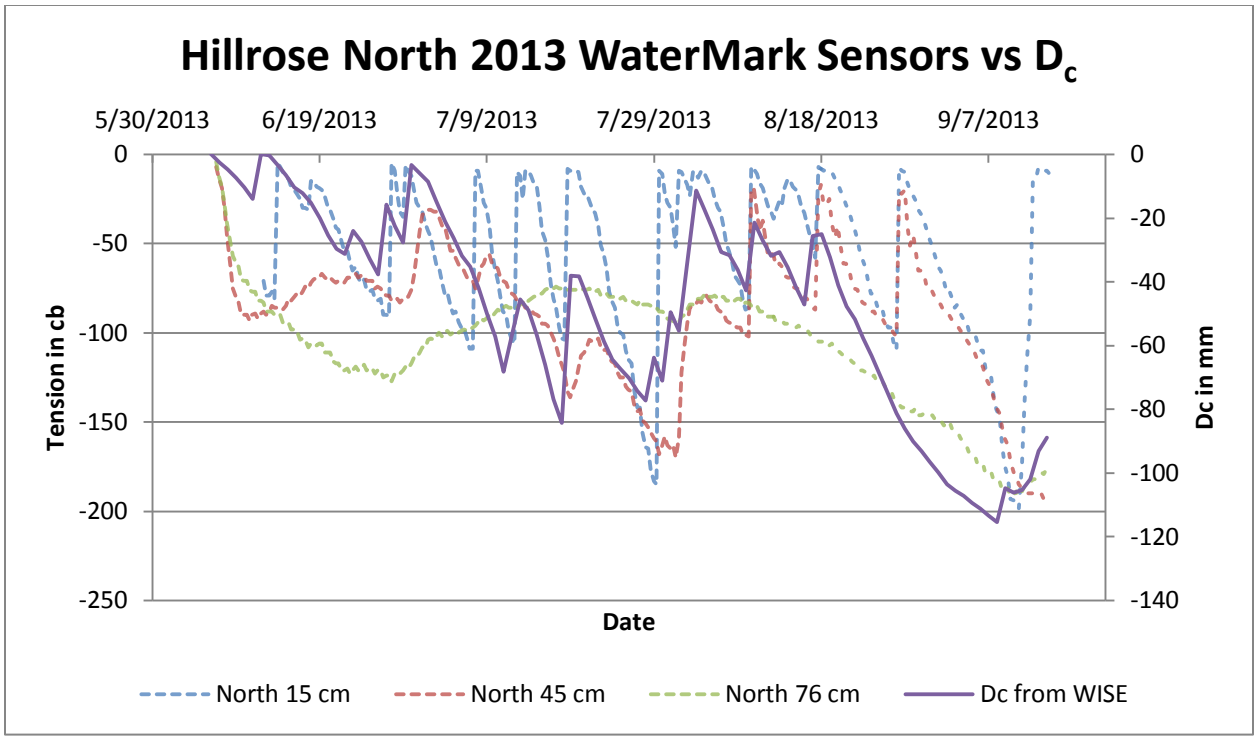


Figure 2.7 – WaterMark data recorded every 8 hours superimposed over D_c (mm) from WISE. Both data sets have similar trends suggesting the tool is working properly.

Table 2.6 – Calculated water balance components using actual irrigations for center pivot irrigated sugar beets at Gilcrest, CO (13 June - 2 September) in 2013 and Gilcrest North 2014 (4 June - 4 September).

Water balance component	Gilcrest South 2013	Gilcrest North 2014
ETc (mm)	398	604
Gross Irr (mm)	424	549
P (mm)	20	165
DP + SRO + irr eff	43	105
ΔS (mm)	3	160

ΔS = change in soil water storage in managed root zone (91 cm)

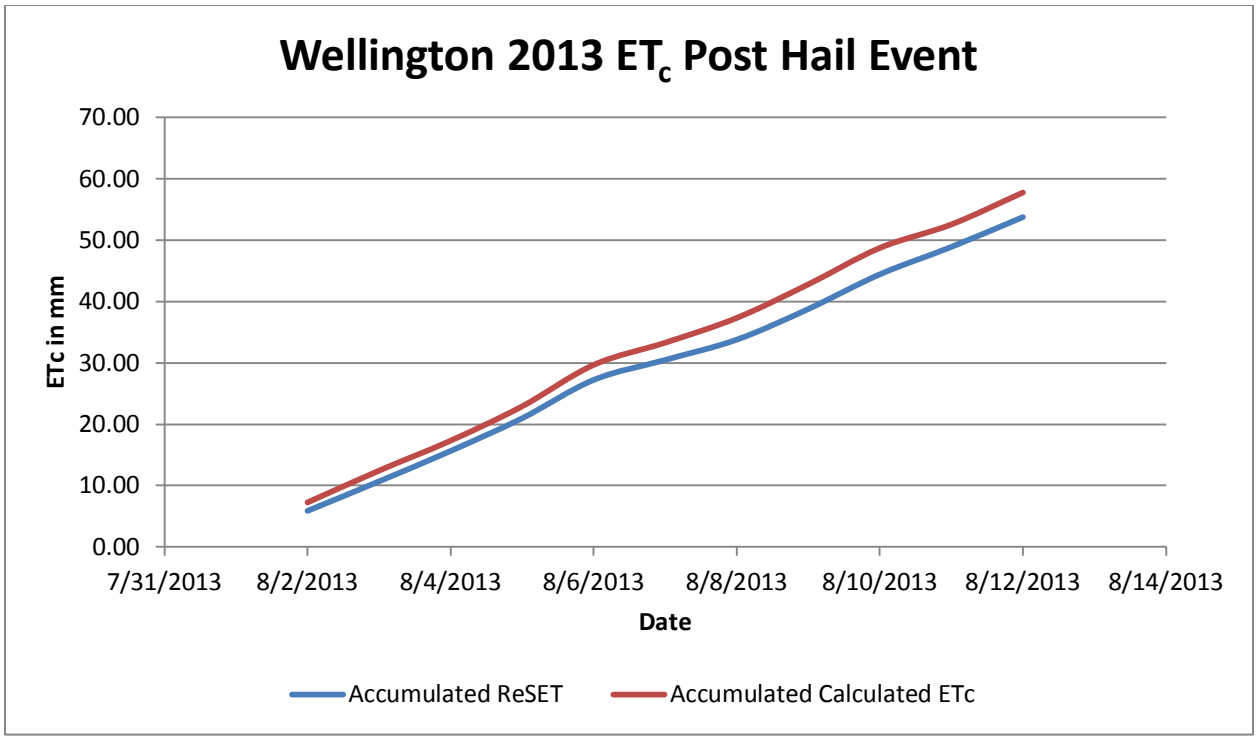


Figure 2.8 – Differences in ReSET and crop coefficient derived ET_c begin to accumulate after a hail event which caused leaf defoliation. Total accumulations after 11 days was 31.52 mm less for the ReSET model.

Table 2.7 -- Comparison of precipitation accumulation from weather stations vs. on-site tipping bucket rain gauge data.

Field-Site	N[a]	Total Difference[b] (mm)	N[a]	Total Difference[b] (mm)
	2013		**2013 Prior to September**	
Gilcrest	137	117.35	102	68.83
Hillrose	123	63.81	118	62.84
Vernon	154	72.39	118	4.06
Wellington	142	9.65	107	6.60
	2014			
Gilcrest	119	12.70		
Hillrose	105	30.56	> Weather Stations	
Vernon	105	46.48	< Weather Stations	
Wellington	100	70.10		

[a]N = number of days; [b]Total Difference = total difference between on-site rain gauge and weather station precipitation data.

REFERENCES

ASCE-EWRI. 2005. The ASCE Standardized Reference Evapotranspiration Equation. Report 0-7844-0805-X, *ASCE Task Committee on Standardization of Reference Evapotranspiration*. Reston, Va.: American Soc. Civil Engineers.

Allen, R. 1996. "Assessing Integrity of Weather Data for Reference Evapotranspiration Estimation." *J. Irrig. Drain Eng.*, 122(2), 97–106.

Allen, R.G., Pereira, L.S., Raes, D., Smith, M., 1998. Crop evapotranspiration – Guidelines for computing crop water requirements – *FAO Irrigation and drainage paper 56*. Food and Agriculture Organization of the United Nations, Rome, Italy: FAO.

Allen, R.G., Wright, J.L., Pruitt, W.O., Pereira, L.S., Jensen, M.E., 2007. Water Requirements. In: *Design and Operation of Farm Irrigation Systems. 2nd ed*. St. Joseph, MI. American Society of Agricultural and Biological Engineers.

Al-Kaisi, M.M., Broner, I., 2009. Crop Water Use and Growth Stages. *Colorado State University Extension Service Fact Sheet no. 4.715*. Available online at: http://www.ext.colostate.edu/pubs/crops/04715.html (verified 5 Nov. 2014).

Andales, A.A., Bauder, T.A., Arabi, M. 2014. A Mobile Irrigation Water Management System Using a Collaborative GIS and Weather Station Networks. In: Ahuja, L.R., Ma, L., Lascano, R., *Applications of Agricultural System Models to Optimize the Use of Limited Water, Volume 6*. (In Press).

Andales, A.A. Chávez, J.L., Bauder T.A., 2011. Irrigation Scheduling: The Water Balance Approach. *Colorado State University Extension Service Fact Sheet no. 4.707*. Available online at: http://www.ext.colostate.edu/pubs/crops/04707.html. (verified 5 Nov. 2014).

Broner, I. (2005). Irrigation Scheduling. Fort Collins: *Colorado State University Extension.*

CoAgMet, 2014. *The Colorado Agricultural Meterological Network,* Available at: http://www.coagmet.com/ accessed [06/05/13]

Grantz, D.A., X.J. Zhang, P.D. Metheney, D.W. Grimes, Indirect measurement of leaf area index in Pima cotton (Gossypium barbadense L.) using a commercial gap inversion method, *Agricultural and Forest Meteorology, Volume 67, Issues 1–2*, December 1993, Pages 1-12, ISSN 0168-1923. Available online at: http://dx.doi.org/10.1016/0168-1923(93)90046-K. (verified 5 Nov. 2014).

Doorenbos, J., Pruitt, W.O., 1977. Guidelines for prediction of crop water requirements. *FAO Irrigation and Drain. Paper No. 24 (revised)*, Rome, Italy: United Nations FAO.

Elhaddad, A. (2014). [ReSET ETc Values]. Unpublished raw data.

Elhaddad,A., L.A. Garcia, and J.L. Chávez. 2011. Using a surface energy balance model to calculate spatially distributed actual evapotranspiration. *J. Irrig. Drain. Eng. 137(1)*:17-26.

Gee, G. W. and D. Or. 2002. Particle-size analysis. pp. 255-293. In: J. Dane and G. Topp (Eds.), *Methods of Soil Analysis*, Part 4, *Physical Methods*. Soil Science Society of America, Madison, WI.

Gleason, D.J. 2013. Evaporation-based Irrigation Scheduling Tools For Use In Eastern Colorado. Master's Thesis, Colorado State University, Fort Collins. Available online at: http://hdl.handle.net/10217/79053 (verified 5 Nov. 2014).

Gleason, D. J., Andales, A. A., Bauder, T. A., & Chávez, J. L. (2013). Performance of atmometers in estimating reference evapotranspiration in a semi-arid environment. *Agricultural Water Management, 130*27-35. doi:10.1016/j.agwat.2013.08.008

Grossman, R.B., Reinsch, T.G., 2002. Bulk Density and Linear Extensibility. pp. 201-228. In: J. Dane and G. Topp (Eds.), *Methods of Soil Analysis*, Part 4, *Physical Methods*. Soil Science Society of America, Madison, WI.

Holen, Carlyle D., Dexter, Alan G. 1997. A Growing Degree Day Equation For Early Sugarbeet Leaf Stages. Published in the *1996 Sugarbeet Extension Reports, Vol 27*: pages 152-157. Sugarbeet Research and Extension Board of Minnesota and North Dakota.

J.M. Welles, J. N. (1991). Instrument for Indirect Measurement of Canopy Architecture. *Agronomy Journal* , 818-825.

Merriam, J. L. (1966). A Management Control Concept for Determining the Economical Depth and Frequency of Irrigation. *Transactions of ASAE*, 492-498.

Natural Resources Conservation Service (NRCS). 2004. Estimation of direct runoff from storm rainfall (Chapter 10). In NRCS. *National Engineering Handbook: Part 630 Hydrology*. USDA, Washington DC. (Available online at ftp://ftp.wcc.nrcs.usda.gov/wntsc/H&H/NEHhydrology/ch10.pdf. (verified 5 Nov. 2014).

Northern Water, 220 Water Avenue, Berthod, CO 80513. http://www.northernwater.org. Date accessed [4/13 – 11/14]

Rachel Barta, I. B. (2004). Colorado High Plains Irrigation Practices Guide. Fort Collins: Colorado Water Resources Research Institute.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly spaced data. ACM National Conference, (pp. 517-524).

Shrestha, N., Geerts, S., Raes, D., Horemans, S., Soentjens, S., Maupas, F., & Clouet, P. (2010). Yield response of sugar beets to water stress under Western European conditions. *Agricultural Water Management, 97*(2), 346-350. doi:10.1016/j.agwat.2009.10.005

Smith, R.E., Warrick, A.W., 2007. Soil Water Relationships. In: *Design and Operation of Farm Irrigation Systems. 2nd ed.* St. Joseph, MI. American Society of Agricultural and Biological Engineers.

Soil Survey Staff. Natural Resource Conservation Service, United States Department of Agriculture. *Web Soil Survey*. Available online at http://websoilsurvey.nrcs.usda.gov/. Accessed [5/14/13]

Topp, G.C., Ferré, P.A., 2002. Water Content. pp. 417-545. In: J. Dane and G. Topp (Eds.), *Methods of Soil Analysis*, Part 4, *Physical Methods*. Soil Science Society of America, Madison, WI.

USDA Staff. (2012, 10 12). U.S. Sugar Production. Retrieved 12 12, 2013, from *USDA Economic Research Service*: http://www.ers.usda.gov/topics/crops/sugar-sweeteners/background.aspx#.UqngISc15AC. (verified 5 Nov. 2014).

Wright J.L. 1982. New evapotranspiration crop coefficients. *J. Irrig. Drain. Div.,* ASCE 108: 57-74

CHAPTER 3: AN IPHONE APP TO EXTEND USE OF A CLOUD-BASED IRRIGATION SCHEDULING TOOL

SYNOPSIS

Irrigation in Colorado, a headwaters State, is crucial for viable agricultural production; consequently, with the foreseen population growth, there will become a greater demand placed on precious water resources. Technology must be adopted and embraced as part of the solution to water shortage. Researchers at Colorado State University have created an online evapotranspiration-based irrigation scheduling tool called Water Irrigation Scheduling for Efficient Application (WISE) that uses the soil water balance method and data queries from Colorado Agricultural Meteorological Network (CoAgMet) and Northern Colorado Water Conservation District (NCWCD) weather stations. To expedite and mobilize required user interaction with the software interface, an iPhone app has been developed that allows users to quickly view their soil moisture deficit, weather measurements, and the ability to input applied irrigation amounts into WISE. Potential users: agricultural producers, irrigation managers, and research scientists will benefit from this app as it allows lite access to the tool from any location within a cellular data network. Technology such as the scheduling tool and iPhone app, when adopted within Colorado and the western United States, allow irrigators another tool to better utilize water resources.

INTRODUCTION

Irrigation in the semi-arid western United States, specifically Colorado, allows agricultural producers to supplement the soil moisture profile with surface or ground water to

produce vibrant crops. Within the headwater state of Colorado, water resources are often over allocated and the need for this limited resource is increasing with higher municipal and industrial demand. It is anticipated that by 2030, Colorado's population will increase by 1.8 million from the current population of 5 million (Colorado Water Conservation Board, 2004). It is also estimated that over the next few decades, demand for water will increase by 777,093,558 m^3 (630,000 acre feet) (Colorado Water Conservation Board, 2004). With continued demand for water, agriculture must discover methods to effectively use this precious resource. Irrigation scheduling technology for both field crops and turf grass will continue to provide solutions to these challenges (Bergez et al., 2001; Klocke et al., 2009; Dobbs et al., 2013). In order to provide a step towards improved agricultural water use, technological advances and fundamental irrigation principles has led to the creation of Water Irrigation Scheduling for Efficient Application (WISE) and the accompanying iPhone app. The app is available on the app store at: https://itunes.apple.com/app/id928128681.

DEVELOPMENT OF A WISE MOBILE APPLICATION

Researchers within the Soil and Crop Science and Civil Engineering departments at Colorado State University have created WISE for agricultural producers, irrigation managers, and research scientists (Andales et al., 2014). The tool resides on a cloud based platform of the environmental Risk Assessment and Management System (eRAMS; Arabi, 2011; https://www.erams.com), and uses the soil water balance (SWB) approach to assist users by providing recommended irrigation amounts for individual fields (Andales et al., 2014). Mandatory setup of a field can only be completed on a web browser via a computer. Upon completion of drawing a field boundary from a base map layer or uploaded via a shape file, USDA SSURGO data (http://sdmdataaccess.nrcs.usda.gov/) is downloaded to obtain soil

physical properties and the available water content, both of which are utilized to determine the soil water profile range. Additionally, weather variables required to calculate the ASCE-EWRI (2005) hourly alfalfa reference evapotranspiration (ET_r), which include: solar radiation, air temperature, relative humidity and/or vapor pressure, and wind speed are downloaded from CoAgMet (http://www.coagmet.colostate.edu/) and NCWCD (http://www.northernwater.org/WaterConservation/WeatherandETInfo.aspx) weather stations. Daily precipitation amounts are also downloaded. This SWB method requires user input of crop type, planting date, emergence date (or green up date for alfalfa and winter wheat), and the initial soil water deficit. After the initial setup, proper use of WISE requires updated irrigation amounts and date, unrecorded precipitation amounts and date, and any in season adjustments to the soil moisture deficit percentage. To streamline this process, the iPhone app allows the user to input these three variables using a mobile device.

The desire for a smart phone app was originally discussed at an annual WISE stakeholders meeting. Of the 14 members representing government, industry, and production agriculture, 29% suggested on an open-ended survey to pursue the development of an app. Additionally, 43% stated they would pay for an app as an advanced feature of WISE.

The difficulty of outlining field shapes on a smart phone or tablet prohibits users from using the full version of WISE on those devices; therefore, the mobile version does not possess full capabilities of WISE. Instead, users can easily view their soil moisture profile, add irrigations, precipitation, and observed soil deficit values once the project and field are setup using a web browser on a personal computer.

Multiple views encompass the hierarchy of the iPhone application. Upon validation of the user login information, a view listing irrigation projects associated with the user account is

displayed. Touching a project from the list brings the user to the field list view. Once a field is selected, a summary page with the soil moisture profile values appears (Figure 3.1 – left). The moisture profile gauge represents the limits between field capacity, wilting point, and the managed allowed depletion values in inches. The current deficit of the root zone is presented by the vertical red bar and the underlying blue bar represents remaining available water. A button for adding irrigation amounts is accessible from this view and if pressed, the add irrigation view will be displayed (figure 3.1 – right). The user can enter an irrigation, precipitation, or percent deficit value, select a date, and touch "update" to send the data string to WISE. In the case of furrow irrigation where the amount applied may be uncertain, the user also has the ability to calculate the amount of irrigation applied within the next view by entering the flow rate, acres irrigated, and hours to complete irrigation, and an algorithm will calculate the equivalent depth of irrigation. Another view option presented on the summary page is "Check Yesterday's Weather" for the current field. When touched, the previous day (or day of maturity for previous seasons) weather variables for the crop field are displayed. In the case that the user has selected more than one weather station, an inverse distance weighted algorithm calculates the approximate weather data specific to the field location. Parameters such as ET_r, crop ET (ET_c), temperature, and growing degree days (GDD) are shown. A complete diagram of the app workflow is presented in figure 3.2.

The Restkit library (www.restkit.org) with license (http://apache.org/licenses/LICENSE-2.0) was added behind the scenes in order to provide simplified access to and parsing of JavaScript Object Notation (JSON) strings between the app and the eRAMS server. This library is used in most views including the login and adding irrigation events. Many other solutions and answers to questions were discovered online and deserve acknowledgment for assisting with this

app's development (Apple Developer; iPhoneDevSDK; Paul Hegarty; RayWenderlich; StackOverflow; tuts+; YouTube). A book discussing basic Objective-C Programming principles was also utilized (Hillegass, 2012).

DISCUSSION

This iPhone application will be beneficial for agricultural producers, irrigation managers, and research scientists by providing quick access to irrigation scheduling information generated by a detailed web-based tool (WISE). It is within the foresight of the development team that one of the greatest probable user errors will be failure to input actual irrigation events. Use of the mobile tool has the ability to reduce this by providing a portable and easily accessible platform for updates. For example, a researcher may be applying a variable rate treatment to multiple plots. By uploading irrigations to each plot via their phone while they are in the field, the data entry will be punctual with a higher chance of being consistent. Upon return to their office, the information can be downloaded and saved as a comma-separated values (csv) file. In another instance, a farmer who is physically checking the soil moisture in a field, can open their phone and check the profile gauge to confirm their decision to apply water. Extending further, weather measurements within a specific area is vital to crop progress and adds another valuable data set at the fingertips of a user.

CONCLUSION

Increasing water shortages in the semi-arid western United States, specifically Colorado, due to population growth and drought, increase the need for more effective use of this resource within production agriculture. Colorado State University is providing farmers, irrigation managers, and research scientists a tool that will help provide a solution to this problem, WISE. However, it is also acknowledged that technology must be worthy and time efficient to be adopted by users.

The iPhone app WISE provides an expedited means by which the user can access and upload information. It also has the ability to reduce user error of misplaced irrigation records and timing by providing the ability to enter this information at any time, in any environment with cellular network access. Considering the benefits of the tool's usage, the app has been created to apply an additional layer of ease and an expedited workflow for the end user.

FIGURES

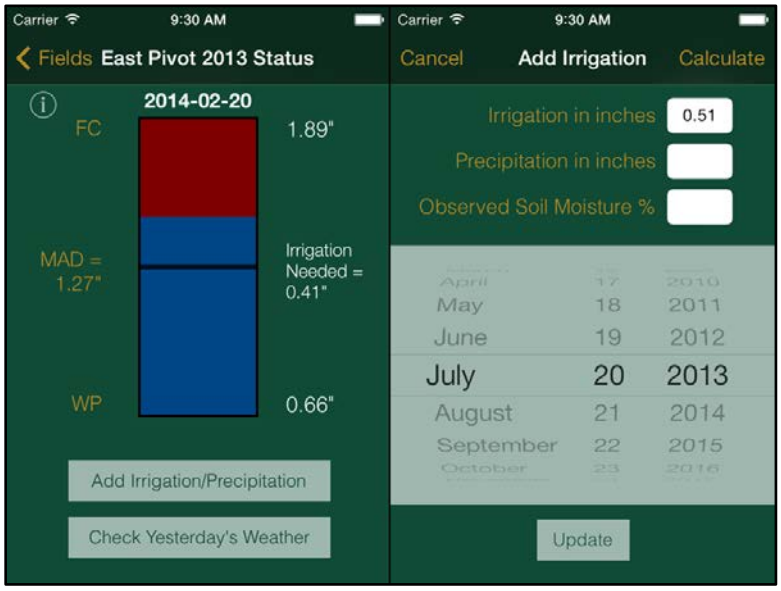


Figure 3.1 – The summary view (left) and the add irrigation view (right). The summary provides a quick access view of the deficit within the soil profile, field capacity, wilting point, managed allowed depletion values as well as a red deficit bar and interactive buttons to proceed to the add irrigation view. To add an irrigation event, the user can input values and a date or continue to the calculation view.

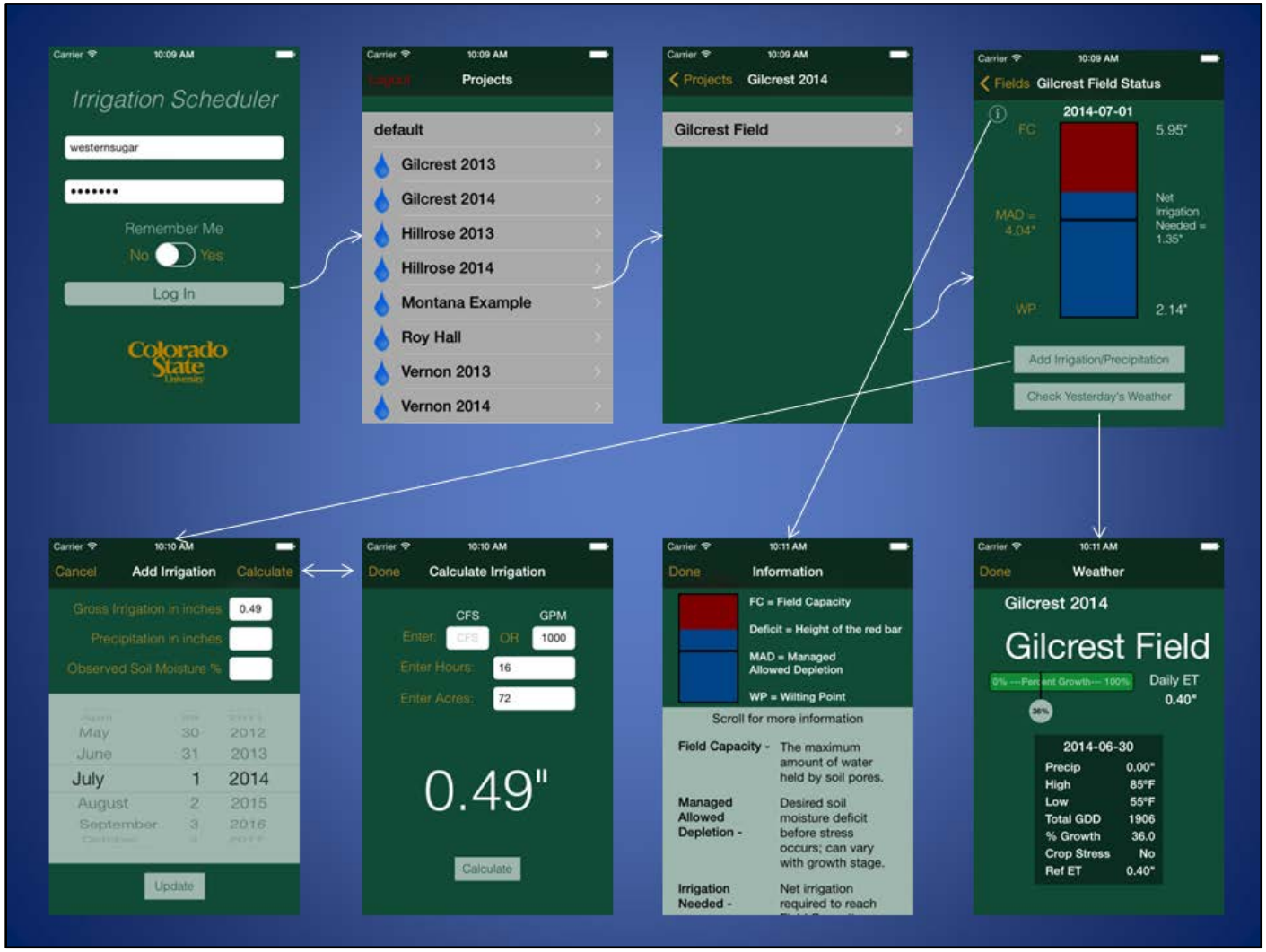


Figure 3.2 – A complete workflow of the app. From left to right top to bottom: login view, project view, field view, summary view, add irrigation view, calculation irrigation amount view, descriptive information view, and weather data view.

REFERENCES

Apple Developer. (n.d.). Retrieved 2014, from https://developer.apple.com/

Andales, A.A., Bauder, T.A., Arabi, M. 2014. A Mobile Irrigation Water Management System Using a Collaborative GIS and Weather Station Networks. In: Ahuja, L.R., Ma, L., Lascano, R. (Eds.), Applications of Agricultural System Models to Optimize the Use of Limited Water, *Advances in Agricultural Systems Modeling Volume 5*. ASA, CSSA, and SSSA, Madison, WI, USA (In Press).

Arabi, M. 2011. Environmental Risk Assessment and Management System. *Colorado Water, Vol. 28, Issue 1*, pp. 15-17. (Available online at http://wsnet.colostate.edu/cwis31/ColoradoWater/Images/Newsletters/2011/CW_28_1.pdf) (verified 5 Nov. 2014).

ASCE-EWRI. 2005. The ASCE Standardized Reference Evapotranspiration Equation. Report 0-7844-0805-X, *ASCE Task Committee on Standardization of Reference Evapotranspiration*. Reston, Va.: American Soc. Civil Engineers.

Bergez, J.E., Debaeke, P., Deumier, J.M., Lacroix, B., Leenhardt, D., Leroy, P., Wallach, D., MODERATO: an object-oriented decision tool for designing maize irrigation schedules, *Ecological Modelling, Volume 137, Issue 1*, 1 February 2001, Pages 43-60.

Colorado Water Conservation Board., 2004. *Statewide Water Supply Initiative.* Denver: CDM.

Dobbs, N., Migliaccio, K., Dukes, M., Morgan, K., and Li, Y. (2013). "Interactive Irrigation Tool for Simulating Smart Irrigation Technologies in Lawn Turf." *J. Irrig. Drain Eng., 139(9)*, 747–754.

Hillegass, A., 2012. *Objective-C Programming The Big Nerd Ranch Guide.* Indianapolis: Pearson Technology Group.

iPhoneDevSDK. (n.d.). Retrieved 2014, from MOBILEDEVHQ: http://iphonedevsdk.com/

Klocke, N., Stone, L., and Bolton, D., 2009. Irrigation Scheduling for Deficit Irrigation. World Environmental and Water Resources Congress 2009: pp. 1-9.

Paul Hegarty, S. U. (n.d.). *iTunesU*. Retrieved 2013, from Developing iOS 7 Apps for iPhone and iPad: https://itunes.apple.com/us/course/developing-ios-7-apps-for/id733644550

RayWenderlich. (n.d.). Retrieved 2014, from Tutorials for Developers and GamersRayWenderlich: http://www.raywenderlich.com/

StackOverflow., 2014. Retrieved 2014, from StackExchange: http://stackoverflow.com/

tuts+. (n.d.). Retrieved 2014, from Tutorials, inspiration and videos to help you learn.: http://code.tutsplus.com/

YouTube. (n.d.). Retrieved 2014, from http://www.youtube.com/

CHAPTER 4: CONCLUSION AND RECOMMENDATIONS

Irrigation is a critical factor in agricultural crop productivity, especially in semi-arid climates. Production agriculture is the largest consumer of freshwater in the United States and increased demands on this resource have stretched the availability to a maximum. In order to better manage freshwater allocations, research about specific plant-water use (transpiration) must be prioritized. Crop coefficient (K_{cr}) curves that convert reference evapotranspiration (ET_r) to actual crop water use (ET_c), allows for crop water usage values to be calculated throughout the growing season.

In the first objective of this thesis, modification of a sugar beet K_{cr} curve for Northeastern Colorado, two cutoff values were adjusted, changing the length of time the crop is in the full canopy vegetative stage which will improve the accuracy of the Water Irrigation Scheduling for Efficient Application (WISE) use with sugar beets. A Colorado Irrigation Scheduler for Annuals (CIS-A) was utilized to predict soil deficit values which were compared against observed deficits. Field and crop attributes were placed in the tool along with irrigation events. Gravimetric soil cores were taken bi-weekly (weekly in 2014) to obtain volumetric water content values, or measured soil deficit values. When comparing predicted versus measured soil deficits, it was found that the original coefficient was significantly overestimating water usage. Cutoff 2 was lengthened from 33% to 43% maturity and cutoff 3 was shortened from 83% to 69% maturity, thus reducing the estimated transpiration of sugar beets in a growing season. These two shifts increased the accuracy of the crop coefficient curve by reducing the relative error over all fields from 76.01% to50.95%. The largest relative error when using the original curve was 299.52% as compared to the lowest error using the modified curve of 0.09%. Reductions in the

relative error from the original to the modified curve prove increased accuracy with these new K_{cr} values.

The second objective of this thesis was to describe the development of a mobile smart phone application for combined use with the cloud based eRAMS irrigation scheduling tool. Due to the difficulty of outlining crop fields via a smartphone or tablet, this app is designed to be used in conjunction with the online tool, thus initial setup is required to be completed on a desktop or laptop computer. The app, specific to the iPhone, does however provide an accessible platform during the growing season. Users can view their soil moisture profile, previous day's weather, upload irrigation and precipitation events, and calculate irrigation amounts in inches from differing flow measurements, acreage, and timing. Providing easy access to this information fosters an environment that encourages users to use this tool and lowers the risk of inputting misinformation. The app is in its first year of release and future adaptation of the tool will be reviewed throughout its usage. The iOS based app is currently available for download at https://itunes.apple.com/app/id928128681.

Recommendations for future related projects include expanding upon the first objective to modify the sugar beet crop coefficient for 55.9 cm (22 inch) row spacing which is more prevalent in Northwest part of the U.S. This will empower the irrigation scheduling tool to encompass a more widespread grower and research user base. Implementing the modification over multiple geographic areas will also allow for a broader spatial adoption of this tool. Both of these recommendations will provide further consistency with the Western Sugar Cooperative and allow for potential future funding. Researching and modifying coefficient curves for other crop species will also open the possibility for other sources of funding as well as increase the diversity of users. It is also recommended to verify the FAO-56 sugar beet root depth. A change in rooting

depth could have huge implications on the soil water deficit per the root zone, thus requiring adjustments to the irrigation requirement.

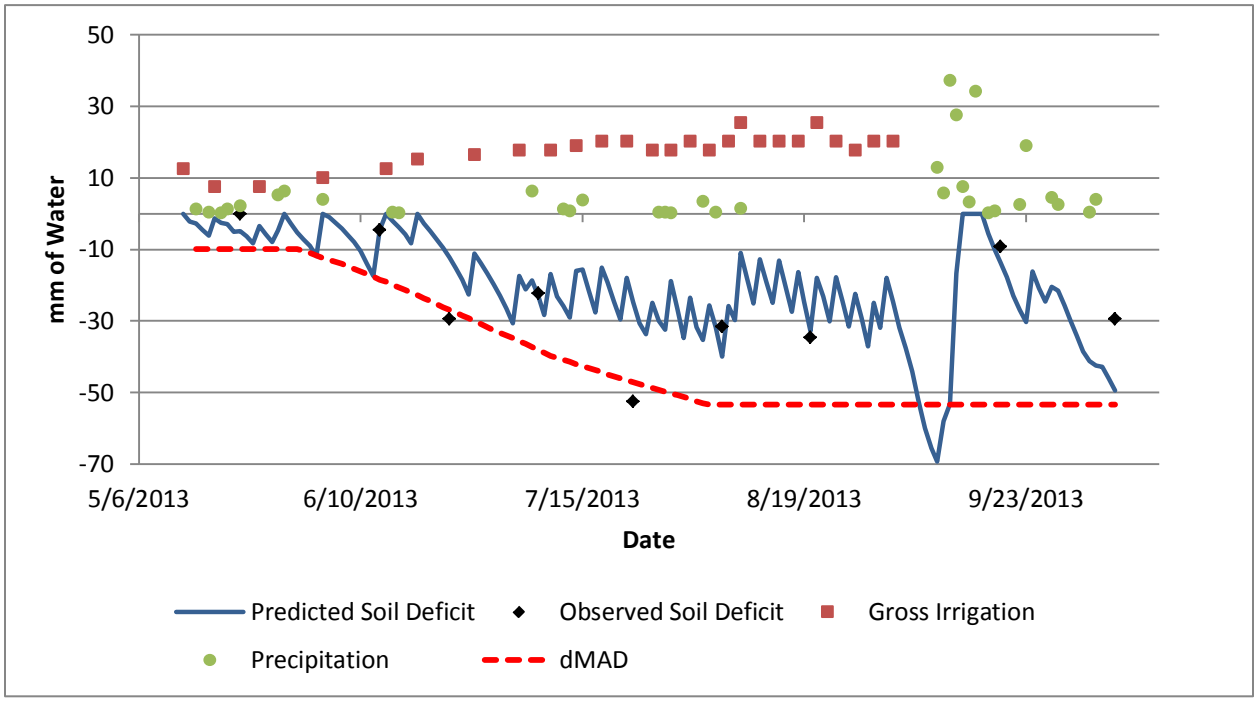
Recommendations for the second objective include further development of the iPhone app. Enabling users the ability to create user accounts, projects, and crop fields will decrease the need for multiple pieces of hardware and possibly increase consistent usage and loyalty towards this tool. Complications such as drawing/outlining a crop field will require special attention as very few applications already possess this ability. This recommendation shadows the trends of modern computing via mobile tablets and phones. Recommendations to both of these objectives will promote usage of the eRAMS irrigation scheduling tool, a software program this thesis has been based around.

In conclusion, within the United States, and especially within Colorado, increased demand for freshwater from growing urban areas, pre-existing state compact requirements, and drought will mean that agriculture will be placed in the spotlight as a possible source of water. This will lead to reduced water availability for irrigation, and will require more efficient use of irrigation water to sustain production. Estimating ET_c and the soil water balance will provide information that can guide irrigation timing and amounts. The modification of the sugar beet K_{cr} will provide more accurate modeling for the sugar beet water usage throughout the growing season. The iPhone application will ensure these values are being relayed to the irrigation scheduling tool and ultimately the users' hands; both of which will allow agriculture to use water allocations more effectively and provide for healthy, vigorous crops.

APPENDIX I

Figures A1 thru A14 – Comparison of the modified crop coefficient predicted soil deficit (blue line) using the soil water balance equation against the observed values (black diamond's). Also shown is the management allowed depletion of the root zone (dMAD; dashed red line), gross irrigation (red squares), and precipitation (green circles). The managed root zone was 711 mm.

Figure A1 – Gilcrest North 2013



Gilcrest South 2013 shown above (Figure 2.5).

Figure A2 – Hillrose North 2013

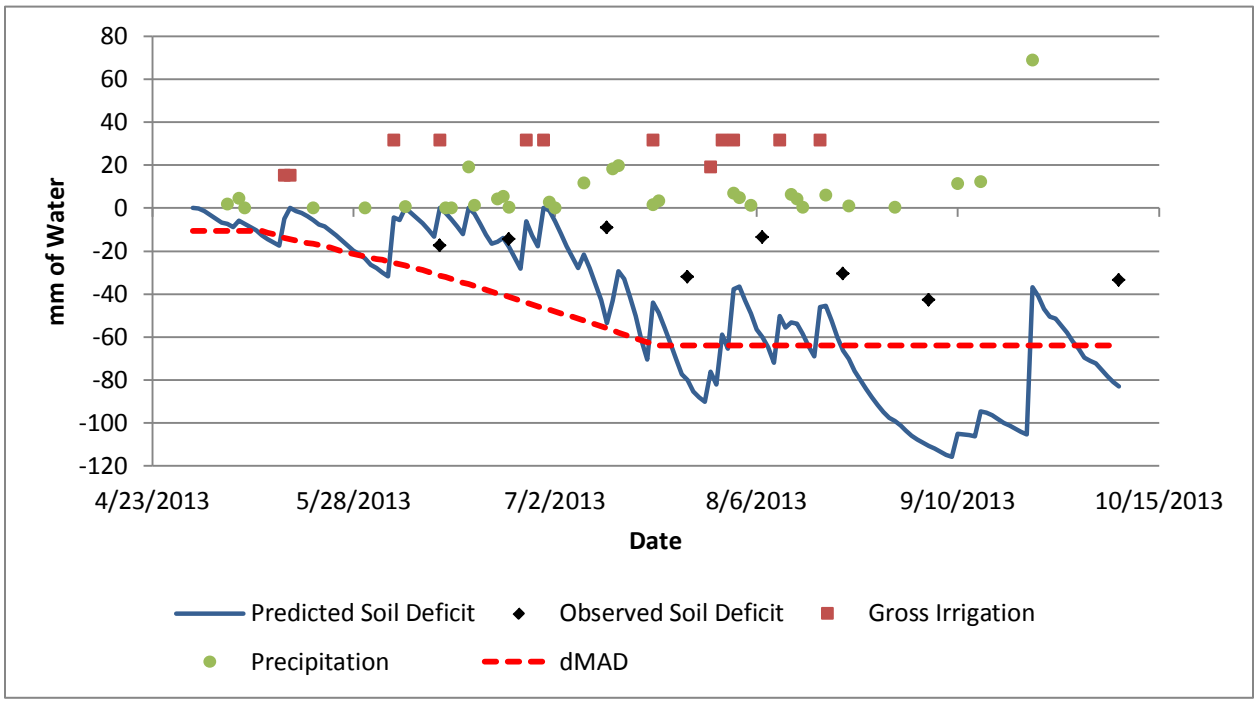


Figure A3 – Hillrose South 2013

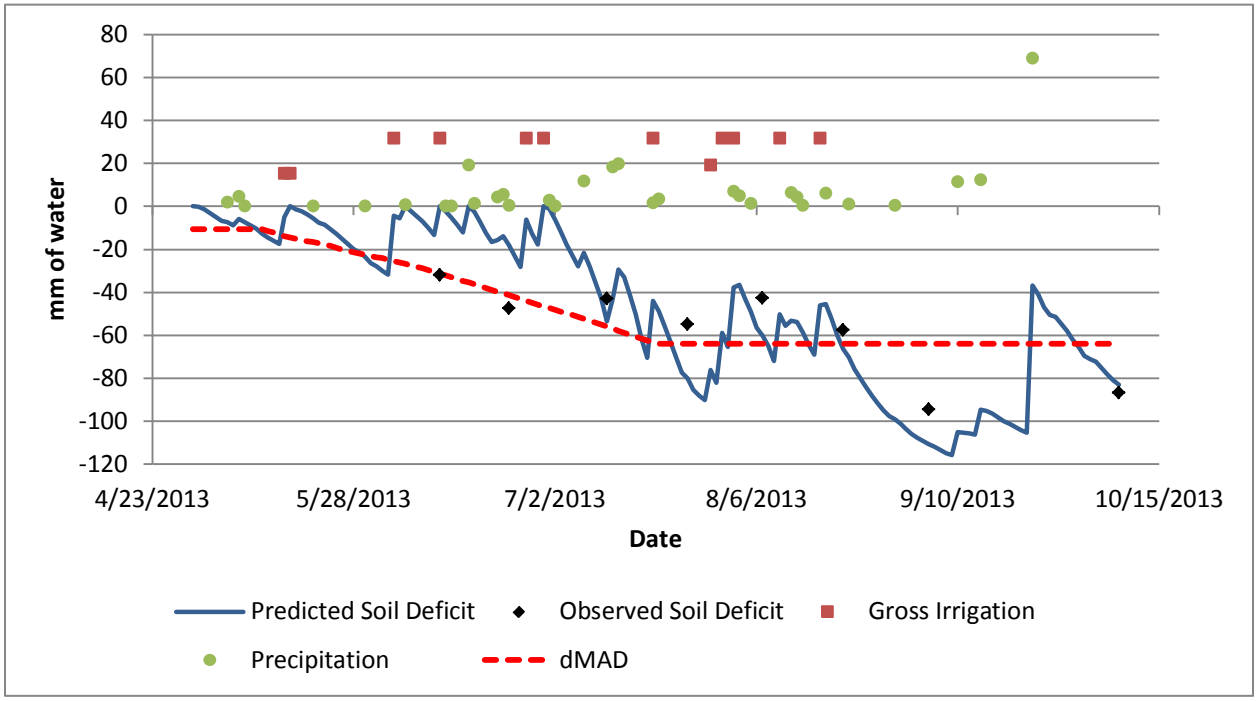


Figure A4 – Vernon East 2013

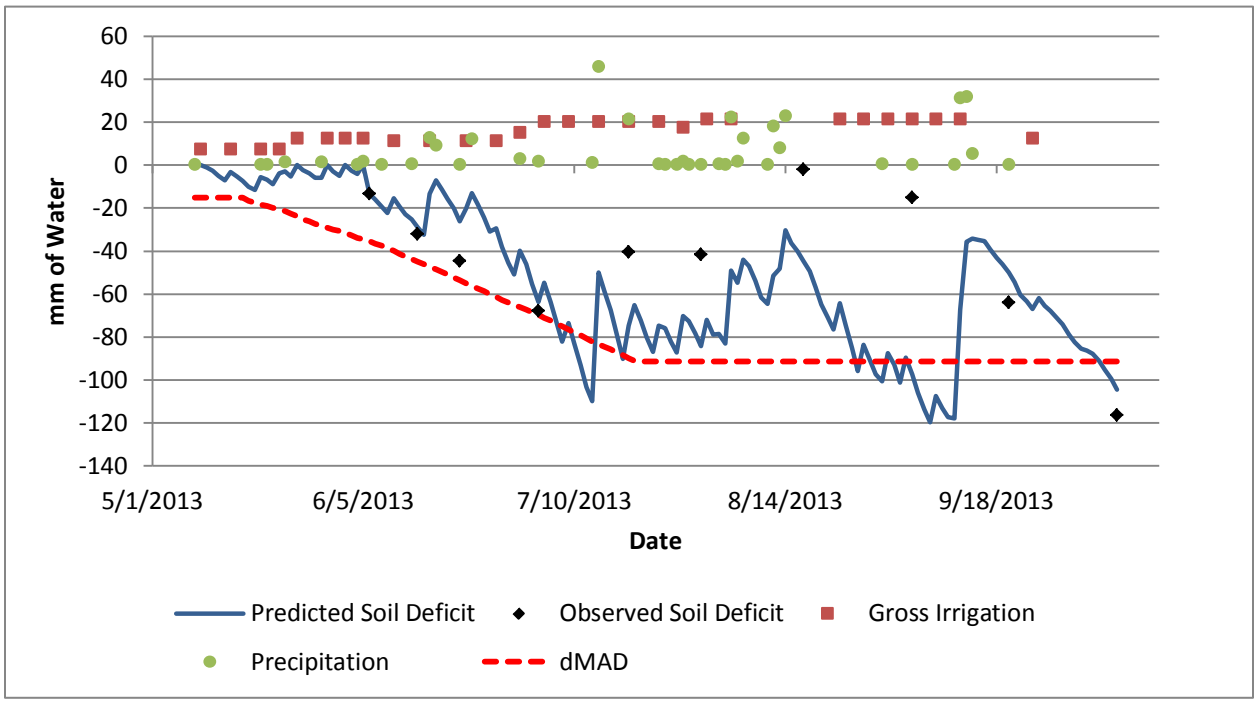


Figure A5 – Vernon West 2013

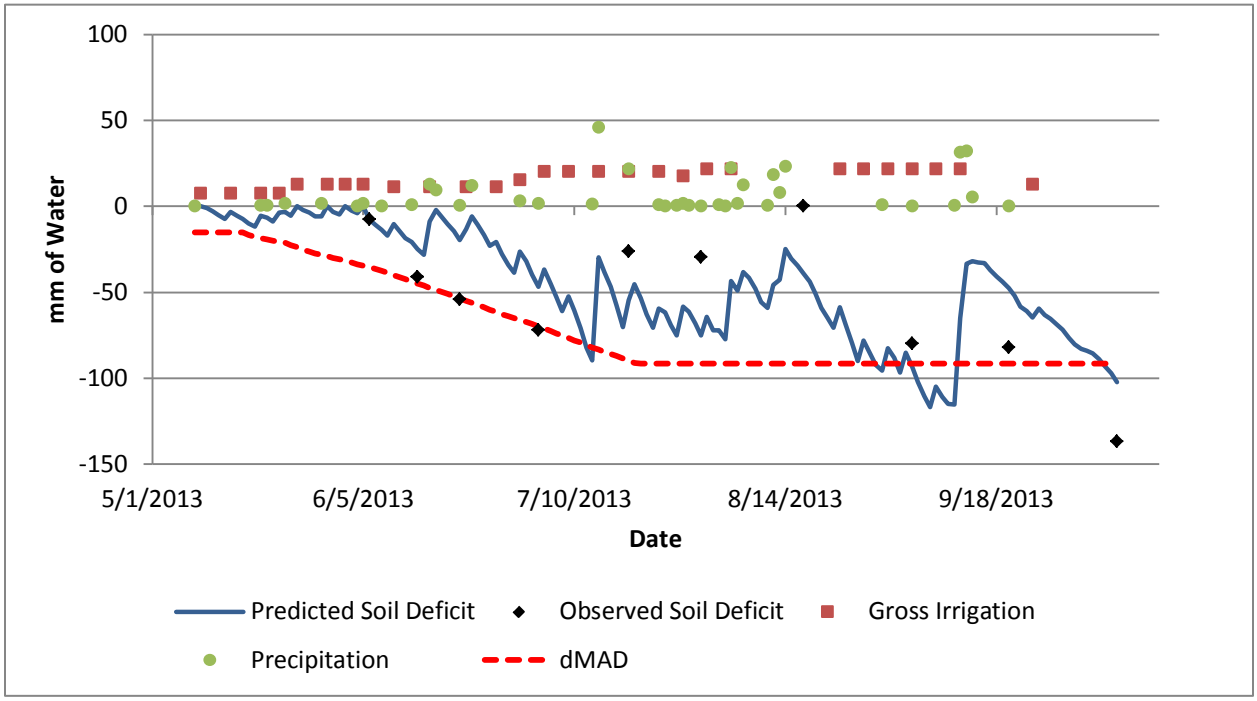


Figure A6 – Wellington North 2013

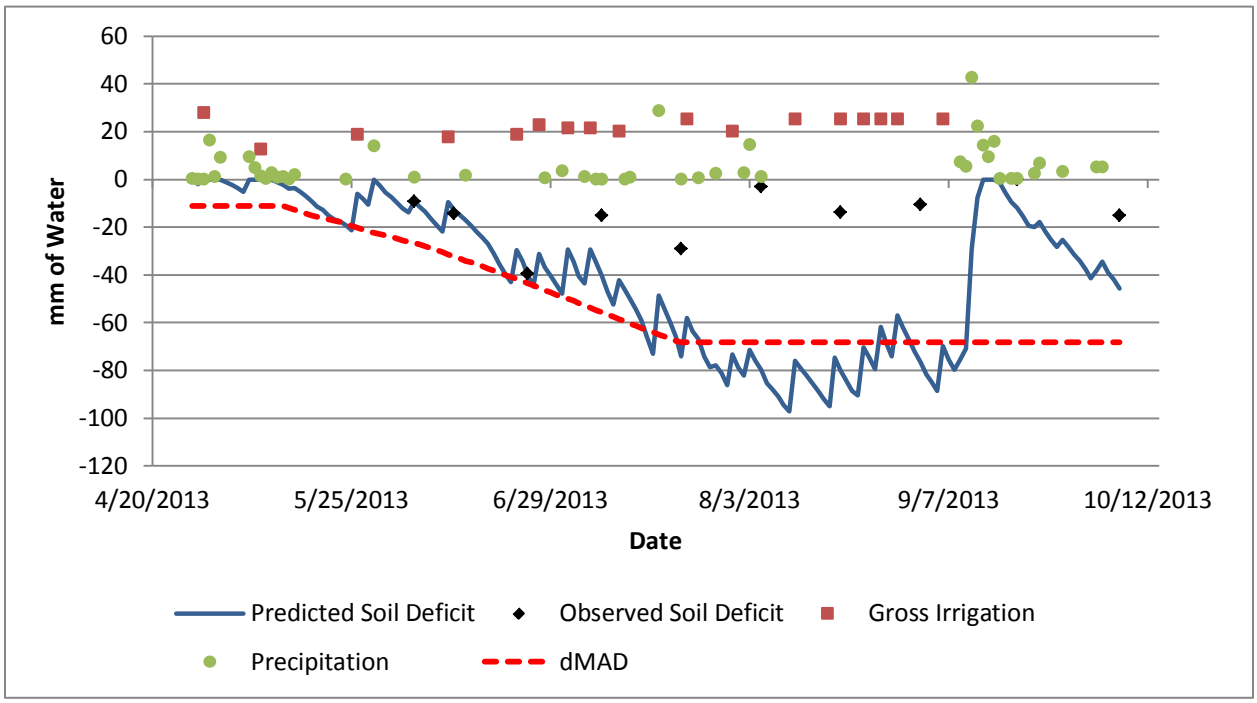


Figure A7 – Wellington South 2013

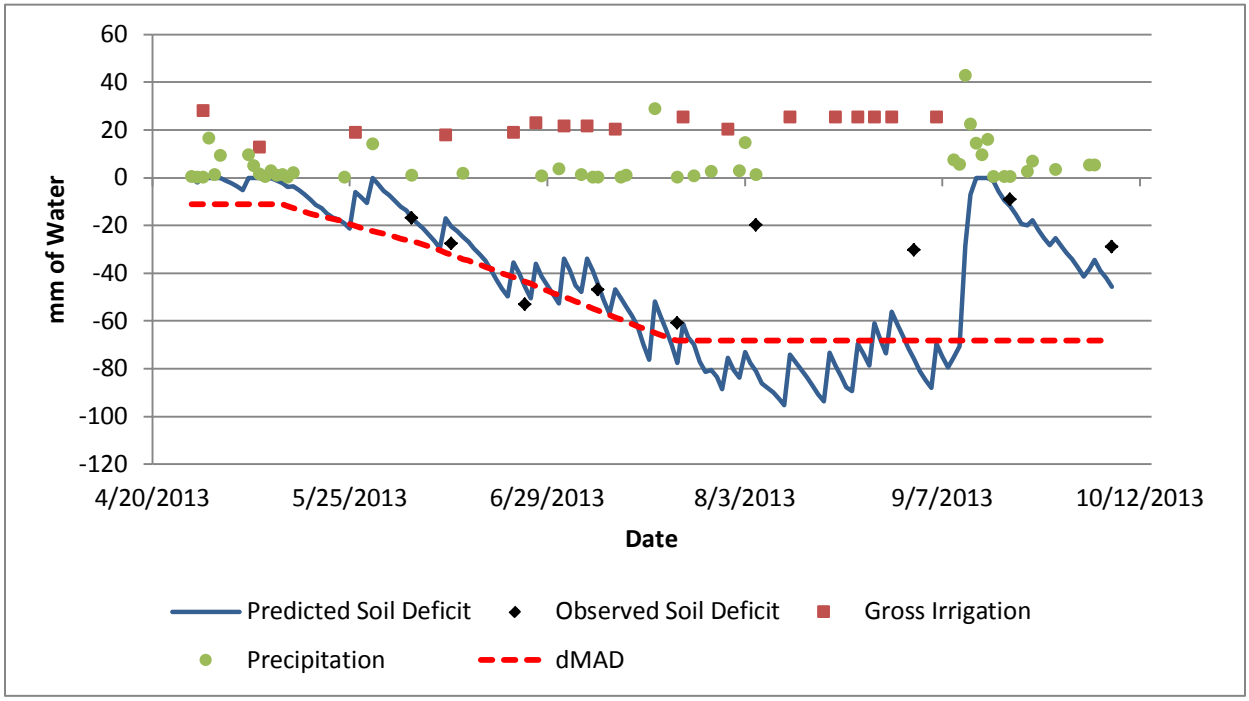


Figure A8 – Gilcrest North 2014

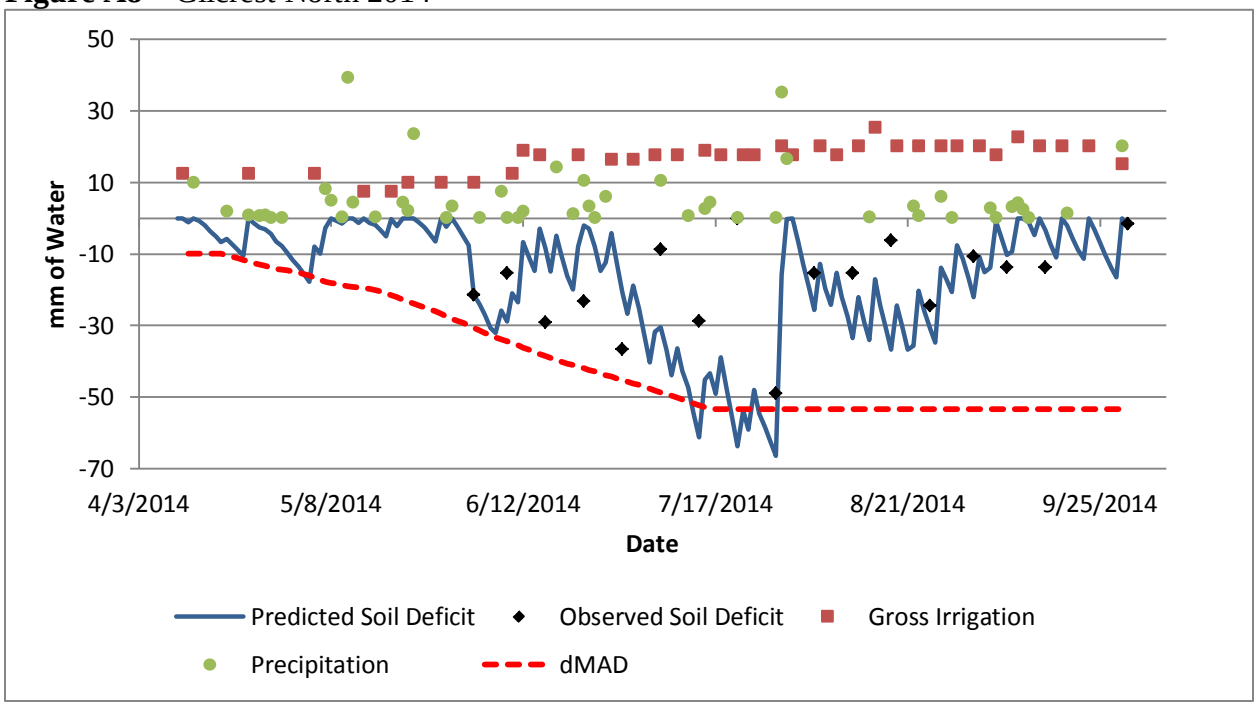
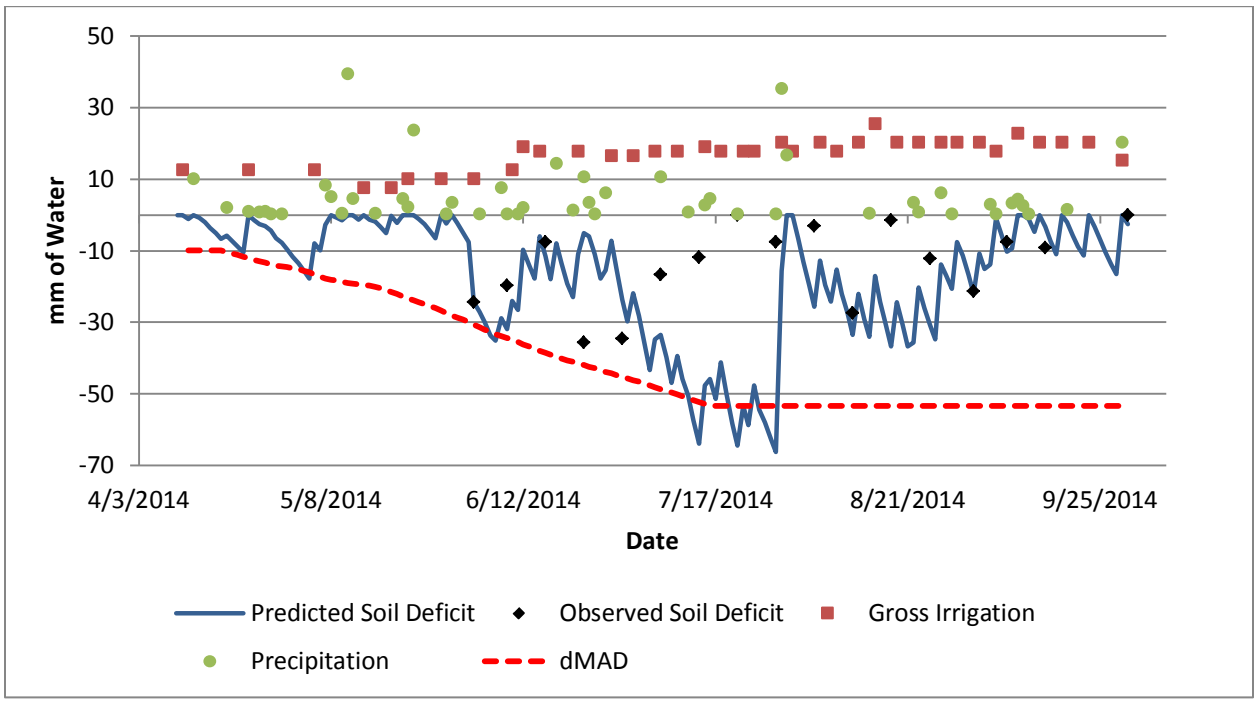


Figure A9 – Gilcrest South 2014



Hillrose East 2014 shown above (Figure 2.6).

Figure A10 – Hillrose West 2014

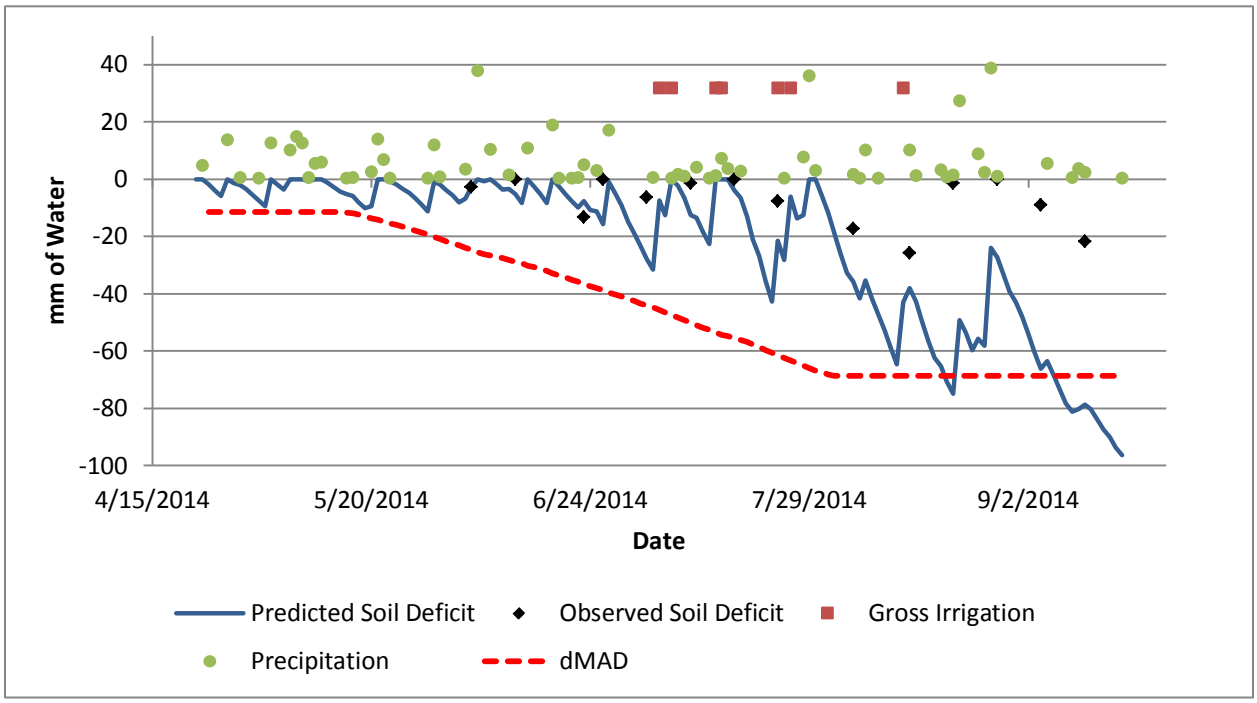


Figure A11 – Vernon East 2014

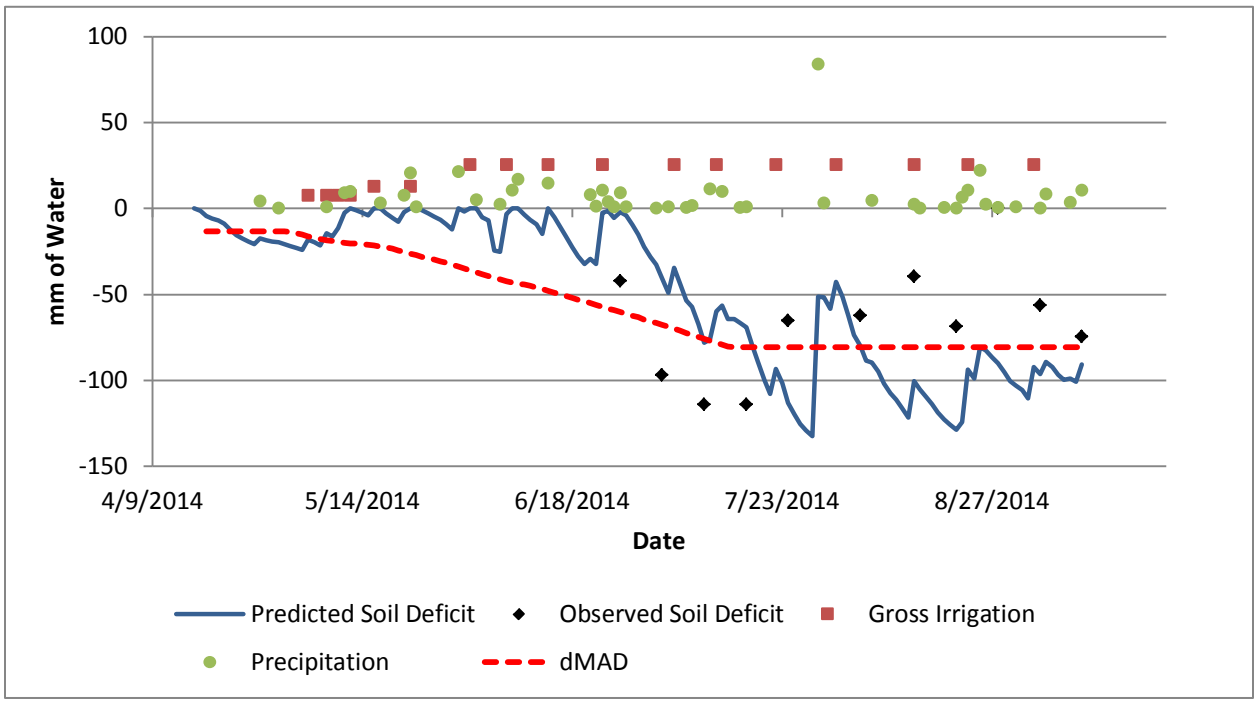


Figure A12 – Vernon West 2014

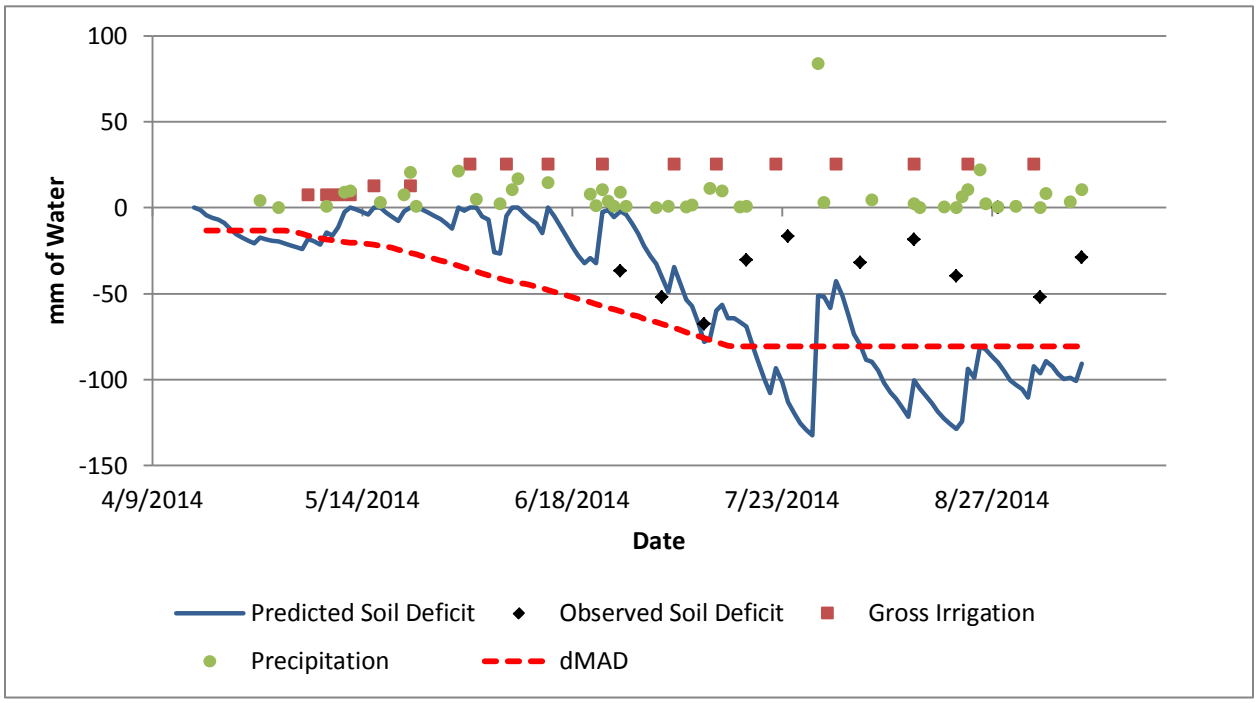


Figure A13 – Wellington North 2014

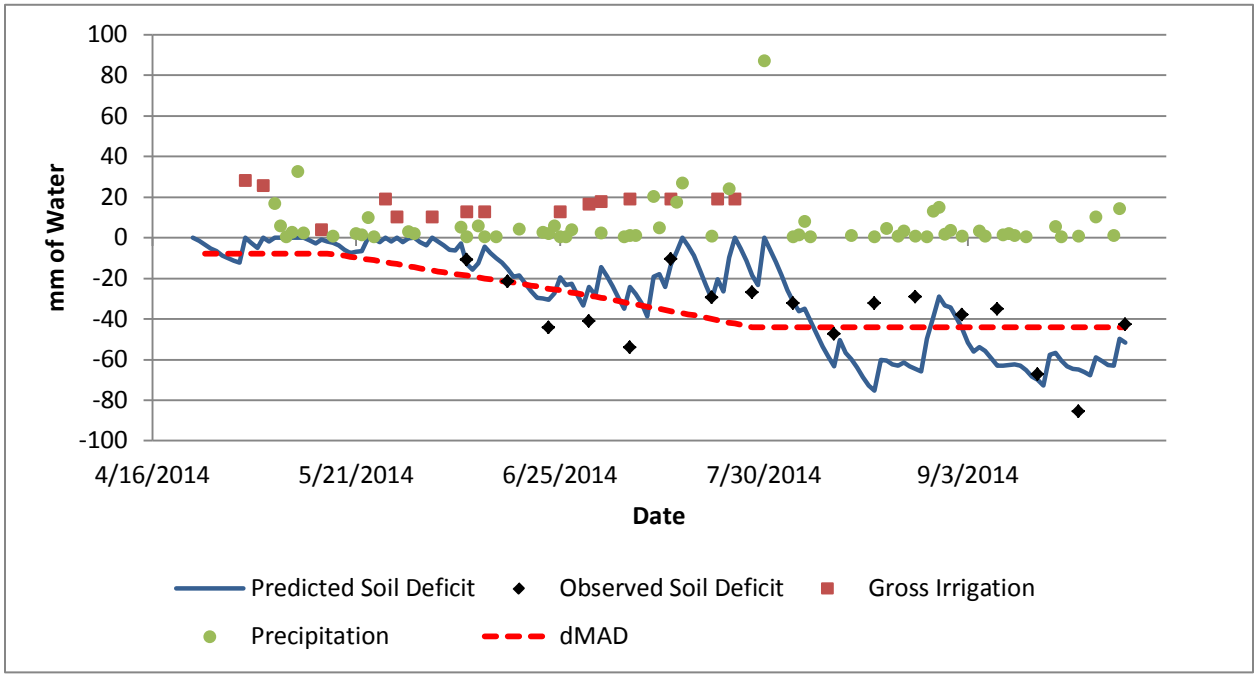
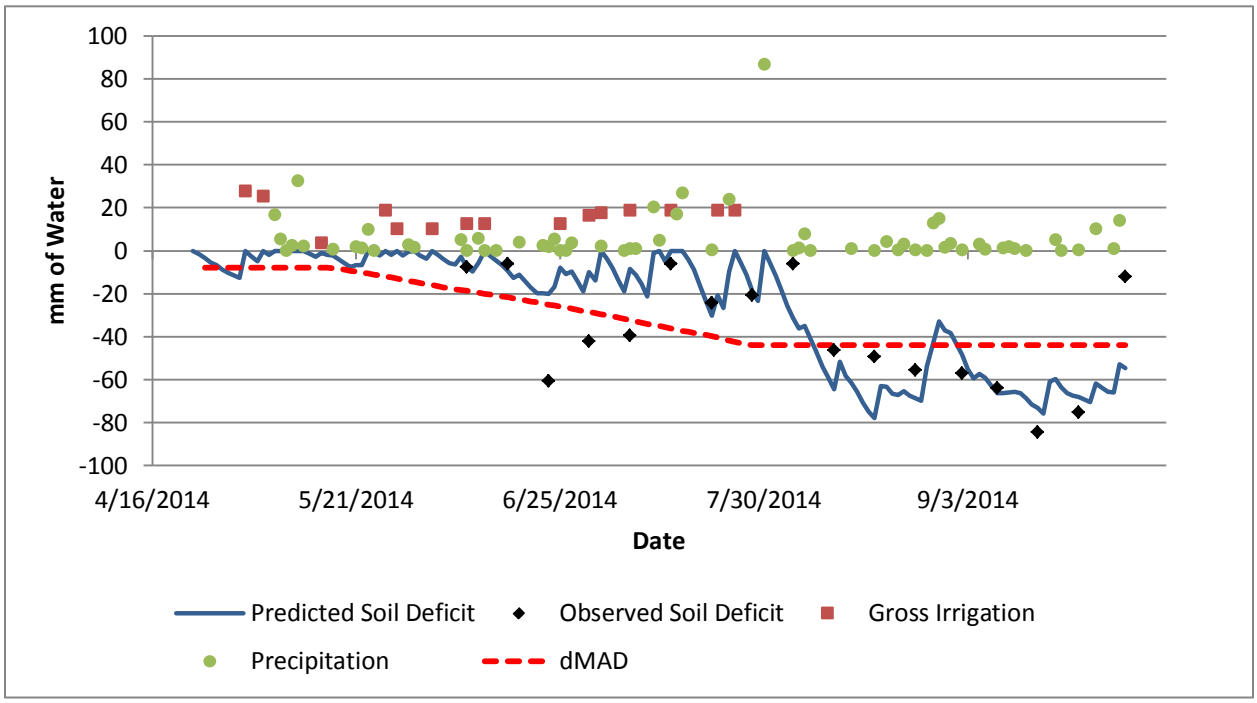


Figure A14 – Wellington South 2014



APPENDIX II

Table A1 – Soil properties for each site. Measurements taken during 2013 and 2014 growing season.

Field-Site	Depth (cm)	Clay%	Sand%	Silt%	Soil Texture	ρb (g cm3)	FC (cm cm-3)	WP (cm cm-3)	AWC (cm cm-3)
2013									
Gilcrest North	0-30	11.67	75.16	13.18	Sandy Loam	1.34	0.20	0.07	0.13
Gilcrest North	30-60	15.00	70.25	14.75	Sandy Loam	1.60	0.24	0.11	0.13
Gilcrest North	60-90	15.00	70.25	14.75	Sandy Loam	1.90	0.30	0.21	0.09
Gilcrest South	0-60	8.33	77.64	14.03	Sandy Loam	1.57	0.23	0.10	0.13
Gilcrest South	30-60	8.33	77.64	14.03	Sandy Loam	1.97	0.25	0.12	0.13
Gilcrest South	60-90	6.67	82.66	10.68	Loamy Sand	1.86	0.22	0.13	0.09
Hillrose North	0-30	32.50	27.70	39.80	Clay loam	1.08	0.26	0.12	0.14
Hillrose North	30-60	40.00	29.50	30.50	Clay	1.39	0.34	0.20	0.14
Hillrose North	60-90	42.50	30.88	26.62	Clay	1.36	0.30	0.16	0.14
Hillrose South	0-30	30.00	28.99	41.01	Clay loam	1.30	0.38	0.24	0.14
Hillrose South	30-60	40.00	24.75	35.25	Clay	1.61	0.39	0.25	0.14
Hillrose South	60-90	37.50	35.99	26.51	Clay loam	1.26	0.28	0.14	0.14
Vernon North	0-30	17.50	39.65	42.85	Loam	1.37	0.38	0.18	0.20
Vernon North	30-60	15.00	37.95	47.05	Loam	1.34	0.31	0.11	0.20
Vernon North	60-90	10.00	42.39	47.61	Loam	1.17	0.32	0.12	0.20
Vernon South	0-30	17.50	39.65	42.85	Loam	1.44	0.39	0.19	0.20
Vernon South	30-60	15.00	37.95	47.05	Loam	1.37	0.33	0.13	0.20
Vernon South	60-90	10.00	42.39	47.61	Loam	1.27	0.31	0.11	0.20
Wellington North	0-30	25.00	44.31	30.69	Loam	1.44	0.35	0.20	0.15
Wellington North	30-60	25.00	44.31	30.69	Loam	1.49	0.29	0.14	0.15
Wellington North	60-90	17.50	62.44	20.06	Sandy Loam	1.59	0.13	-0.02	0.15
Wellington South	0-60	25.00	40.69	34.31	Loam	1.59	0.42	0.27	0.15
Wellington South	30-60	25.00	40.69	34.31	Loam	1.72	0.40	0.25	0.15
Wellington South	60-90	22.50	41.83	35.67	Loam	1.59	0.35	0.20	0.15

2014*									
Gilcrest	0-90	8.33	80.62	11.05	Loamy Fine Sand	1.58	0.18	0.05	0.12
Hillrose	0-75	40.00	25.48	34.52	Clay	1.40	0.40	0.25	0.15
Hillrose	75-90	22.50	59.92	17.58	Sandy Clay Loam	1.53	0.38	0.23	0.15
Vernon	0-90	17.50	43.85	38.65	Loam	1.27	0.28	0.11	0.17
Wellington	0-90	27.50	49.69	22.81	Sandy Clay Loam	1.50	0.26	0.16	0.10

*Composite samples from field site.

APPENDIX III

W.I.S.E. iOS CODE

(Excluding Restkit library)
(Classes separated by horizontal line)

Text color identification:
Green = Comments, documentation comments
Red = Strings
Brown = Preprocessor statements, project preprocessor macros, attributes
Purple = Other class names, other function and method names, other constants, other type names, other instance variables and globals
Black = Plain text
Pink = Keywords
Blue = Characters, numbers
Light blue = URL's
Teal = Project class names, project function and method names, project constants, project type names, project instance variables and globals

```
//  Created by Andrew Bartlett on 3/5/13.
//  Copyright (c) 2013 edu.ColoradoStateUniversity All rights reserved.
```

```
//  WaterLoginViewController.h

#import <UIKit/UIKit.h>

@interface WaterLoginViewController : UIViewController <UITextFieldDelegate> {
    // Save login, yes or no?
    IBOutlet UISwitch *onOffSwitch;
}

@property (strong, nonatomic) IBOutlet UITextField *userName;
@property (strong, nonatomic) IBOutlet UITextField *passWord;
@property (strong, nonatomic) IBOutlet UIButton*loginButton;

// Logout button
- (IBAction)logout:(UIStoryboardSegue *)segue;
// Login button
- (IBAction)loginButton:(id)sender;

// Spinner = loading indicator
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *spinner;

@end
```

```
//  WaterLoginViewController.m

#import "WaterLoginViewController.h"
#import "WaterProjectViewController.h"
#import <QuartzCore/QuartzCore.h>

//Headers to use Restkit
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
#import <SystemConfiguration/SystemConfiguration.h>
#import <MobileCoreServices/MobileCoreServices.h>

#import <RestKit/RestKit.h>
#import "RKProjects.h"
#import "RKFields.h"

#import "RKEramsUser.h"

#import "WaterProjectViewController.h"

@interface WaterLoginViewController ()

@end

@implementation WaterLoginViewController
@synthesize userName, passWord, loginButton, projectString, projectListToTransfer,
spinner;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    // Round button edges
    // Need to import quartz
    loginButton.layer.cornerRadius = 5;
```

```
    loginButton.clipsToBounds = YES;

    //Change status bar text color
    [[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleLightContent];

    self.navigationController.navigationBar.tintColor = [UIColor blackColor];

    // Show alert on initial startup to create field on a computer
    if (![[NSUserDefaults standardUserDefaults] boolForKey:@"HasLaunchedOnce"]) {
        //first launch
        // This will show an alert telling the user to start the process using eRams
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle: @"Irrigation Scheduler"
                                                        message:@"Before you can use the most advance
Irrigaiton Scheduling App, please create your field on a computer using eRams"
                                                       delegate:self
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
        [alert show];
        [[NSUserDefaults standardUserDefaults] setBool:YES
forKey:@"HasLaunchedOnce"];
    } else {
        //app has already launched, no need to show alert
    }

    // Check to see if a username and password were saved on last login
    // Switch must have been set to 'yes' in order to save login info
    NSString *savedUserName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"userNameSaved"];
    NSString *savedPassWord = [[NSUserDefaults standardUserDefaults]
stringForKey:@"passWordSaved"];

    // Check length of saved username and passwords
    NSInteger savedUserNameChar = [savedUserName length];
    NSInteger savedPassWordChar = [savedPassWord length];

    // If username or password length is 0, do nothing
    if (savedUserNameChar == 0 || savedPassWordChar == 0) {
        //do nothing

    } else {
        // If they were saved, put them in the textfields and click the login button for the user
        self.userName.text = savedUserName;
        self.passWord.text = savedPassWord;
        [loginButton sendActionsForControlEvents:UIControlEventTouchUpInside];
    }
}
```

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(void)textFieldDidBeginEditing:(UITextField *)textField
{
    // Black keyboard
    textField.keyboardAppearance = UIKeyboardAppearanceDark;

    // Add labels to the top of the keyboard in order for users to know which textfield they
are within
    UIToolbar* wordToolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0, 320,
50)];
    wordToolbar.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel = [[UIBarButtonItem alloc]
initWithTitle:userName.placeholder style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];

    //Change color of labels to gold
    [textfieldLabel setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    wordToolbar.items = [NSArray arrayWithObjects:textfieldLabel, flexibleSpace,
doneButton, nil];
    [wordToolbar sizeToFit];
    if (textField == userName) {
        userName.inputAccessoryView = wordToolbar;
    }

    UIToolbar* wordToolbar2 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    wordToolbar2.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel2 = [[UIBarButtonItem alloc]
initWithTitle:passWord.placeholder style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace2 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
```

```
    UIBarButtonItem *doneButton2 = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];
    wordToolbar2.items = [NSArray arrayWithObjects:textfieldLabel2, flexibleSpace2,
doneButton2, nil];
    [wordToolbar2 sizeToFit];
    [textfieldLabel2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == passWord) {
        passWord.inputAccessoryView = wordToolbar2;
    }
}

- (void)doneWithNumberPad
{
    // Dismiss keyboard
    [userName resignFirstResponder];
    [passWord resignFirstResponder];
}

// Activates the return button on the keyboard
- (BOOL)textFieldShouldReturn:(UITextField *)theTextField {
    if (theTextField == self.userName) {
        //[theTextField resignFirstResponder]; // This would dismiss keyboard
        // If in username, click 'next' to go to password text field
        [self.passWord becomeFirstResponder];
    }

    if (theTextField == self.passWord) {
        // if in password, click 'go' to login
        [loginButton sendActionsForControlEvents:UIControlEventTouchUpInside];
    }
    return YES;
}

// Activates logout button
- (IBAction)logout:(UIStoryboardSegue *)segue
{
    // if logout button is pressed, erase most saved data

        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"userNameSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"passWordSaved"];

        [[NSUserDefaults standardUserDefaults] setObject:@"" forKey:@"tokenName"];
```

```
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"currentProjectID"];

        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"cropTypeSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"irrigationMethodSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"fieldNameSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"emergenceDateSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"irrigationTypeSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"irrigationFrequencySaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"irrigationEfficiencySaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@""
forKey:@"plantingDateSaved"];
        [[NSUserDefaults standardUserDefaults] setObject:@"" forKey:@"ProjectID"];

        [self dismissViewControllerAnimated:YES completion:NULL];

        //Also need to erase memory of username and password if switch was set to no
        NSString *loginToken = [[NSUserDefaults standardUserDefaults]
stringForKey:@"userNameSaved"];
        NSString *projectID = [[NSUserDefaults standardUserDefaults]
stringForKey:@"passWordSaved"];
        NSLog(@"username, password after logging out is %@, %@", loginToken,
projectID);

        // Textfield will not show password, but username will be visible until application is
quit
        passWord.text = @"";

        // Send user a 3 second message that they successfully logged out
        // User can dismiss message before 3 seconds are expired
        UIAlertView *logout = [[UIAlertView alloc] initWithTitle: @""
                                                         message:@"You've successfully logged out."
                                                        delegate:self
                                               cancelButtonTitle:@"OK"
                                               otherButtonTitles:nil];
        [logout show];
        [self performSelector:@selector(dismissAlert:) withObject:logout afterDelay:3.0f];
}
```

```
// Have it dismiss the logout message
-(void)dismissAlert:(UIAlertView *) alertView
{
   [alertView dismissWithClickedButtonIndex:nil animated:YES];
}

// loginButton clicked, as user is trying to login
- (IBAction)loginButton:(id)sender
{
   // spinner is the loading indicator
   [spinner startAnimating];

   // If either username and password are empty, send user an error message
   if ([self.userName.text length] > 0 && [self.passWord.text length] > 0){

      // If user selected to save login info, store username and password in
NSUserDefaults
      if (onOffSwitch.on) {
         //save username and password
         [[NSUserDefaults standardUserDefaults] setObject:self.userName.text
forKey:@"userNameSaved"];
         [[NSUserDefaults standardUserDefaults] setObject:self.passWord.text
forKey:@"passWordSaved"];

         NSString *savedUserName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"userNameSaved"];
         NSString *savedPassWord = [[NSUserDefaults standardUserDefaults]
stringForKey:@"passWordSaved"];
         NSLog(@"Username: %@; Password: %@", savedUserName, savedPassWord);
      }

      // Use Restkit Get request to see if a token is returned from eRAMS for username
and password
      RKObjectMapping *mapping = [RKObjectMapping
mappingForClass:[RKEramsUser class]];
      // objects on left are variables on eRAMS, objects on right are properties in
RKEramsUser class
      [mapping addAttributeMappingsFromDictionary:@{
                                          @"first_name":   @"firstName",
                                          @"oauth.token":  @"token"
                                          }];

      // Register our mappings with the provider using a response descriptor
      RKResponseDescriptor *responseDescriptor = [RKResponseDescriptor
responseDescriptorWithMapping:mapping method:RKRequestMethodGET
pathPattern:nil keyPath:nil statusCodes:nil];
```

```
        // create our URL request
        NSString *URLString = [[NSString alloc]
initWithFormat:@"https://erams.com/api/authenticate/?username=%@&password=%@
&client_id=a215aa6e-b35b-11e2-a331-000c29f3b793&client_secret=notasecret",
self.userName.text, self.passWord.text];

        // create our URL based on the string
        NSURL *baseURL = [NSURL URLWithString:URLString];

        // perform a request to the server
        NSURLRequest *request = [NSURLRequest requestWithURL:baseURL];
        RKObjectRequestOperation *operation = [[RKObjectRequestOperation alloc]
initWithRequest:request responseDescriptors:@[ responseDescriptor ]];
        [operation setCompletionBlockWithSuccess:^(RKObjectRequestOperation
*operation, RKMappingResult *result) {
            // we have successfully retrieved data to *result
            NSLog(@"array: %@", [result array]);
            // Read the saved loginToken to NSUserDefaults - will use in many other requests
            NSString *loginToken = [[NSUserDefaults standardUserDefaults]
stringForKey:@"tokenName"];

            NSLog(@"LoginToken is %@", loginToken);

            // If eRAMS returns a token, go to the next method
            if ([loginToken length] > 0) {
                [self letsGoToTheNextView];
                [spinner stopAnimating];

                // Otherwise, show message
            } else {
                UIAlertView *failedLogin = [[UIAlertView alloc] initWithTitle: @"Login
Failed"
                                                        message:@"Please enter the correct login
information for eRAMS.com"
                                                       delegate:self
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
                // Login failed, stop loading indicator
                [failedLogin show];
                [spinner stopAnimating];
            }

            // Restkit failure
        } failure:^(RKObjectRequestOperation *operation, NSError *error){
            RKLogError(@"My Operation failed with error: %@", error);
```

```
        //This error has been caused by lack of internet access, eRAMS being down,....
        UIAlertView *failedLogin = [[UIAlertView alloc] initWithTitle: @"Login
Failed"
                                                    message:@"Please check your wireless
connection and/or the eRAMS server."
                                                   delegate:self
                                          cancelButtonTitle:@"OK"
                                          otherButtonTitles:nil];
        [failedLogin show];
        [spinner stopAnimating];
    }];
    // Start Restkit GET request, which uses breaks (^)
    [operation start];

  // User didn't enter both username and/or password
  } else {
    UIAlertView *failedLogin = [[UIAlertView alloc] initWithTitle: @"Login Failed"
                                                    message:@"Please enter the correct login
information for eRAMS.com"
                                                   delegate:self
                                          cancelButtonTitle:@"OK"
                                          otherButtonTitles:nil];
    [failedLogin show];
    [spinner stopAnimating];
  }
}


-(void)letsGoToTheNextView{
  // Initiate the project list view controller since we now have the loginToken
  // Identifier is found on the storyboard
  WaterProjectViewController *projectVC = [self.storyboard
instantiateViewControllerWithIdentifier:@"WaterProjectViewNavigationController"];
  [self presentViewController:projectVC animated:YES completion:NULL];
}

@end
```

```
//  RKEramsUser.h

#import <Foundation/Foundation.h>

@interface RKEramsUser : NSObject

// These are all the variables that eRAMS are providing
```

```
@property (nonatomic, copy) NSString *lastName;
@property (nonatomic, copy) NSString *coordsystem;
@property (nonatomic, copy) NSString *modelingareaCurrent;
@property (nonatomic, copy) NSString *id;
@property (nonatomic, copy) NSString *occupation;
@property (nonatomic, copy) NSString *readOnly;
@property (nonatomic, copy) NSString *city;

@property (nonatomic, copy) NSString *firstName;

@property (nonatomic, copy) NSString *tz;
@property (nonatomic, copy) NSString *zipcode;
@property (nonatomic, copy) NSString *access;
@property (nonatomic, copy) NSString *state;
@property (nonatomic, copy) NSString *distunits;
@property (nonatomic, copy) NSString *email;
@property (nonatomic, copy) NSString *username;
@property (nonatomic, copy) NSString *fax;
@property (nonatomic, copy) NSString *needsUpdate;
@property (nonatomic, copy) NSString *isTemp;
@property (nonatomic, copy) NSString *phone;
@property (nonatomic, copy) NSString *stateJson;
@property (nonatomic, copy) NSString *mapscale;
@property (nonatomic, copy) NSString *purpose;
@property (nonatomic, copy) NSString *address;

@property (nonatomic, copy) NSString *token;

@property (nonatomic, copy) NSString *version;
@property (nonatomic, copy) NSString *expires;
@property (nonatomic, copy) NSString *areaunits;
@property (nonatomic, copy) NSString *fontSize;
@property (nonatomic, copy) NSString *bounds;
@property (nonatomic, copy) NSString *previousLogin;
@property (nonatomic, copy) NSString *organization;
@property (nonatomic, copy) NSString *user;

@end
```

```
//  RKEramsUser.m

#import "RKEramsUser.h"

@implementation RKEramsUser
```

```
- (NSString *)description
{
    //Save token to NSUserDefaults
    NSString *currentToken = self.token;
    [[NSUserDefaults standardUserDefaults] setObject:currentToken
forKey:@"tokenName"];
    NSLog(@"RKEramsUser Token is %@", currentToken);

    return [NSString stringWithFormat:@"%@ (token: %@)", self.firstName, self.token];
}

@end
```

```
//  WaterProjectViewController.h

#import <UIKit/UIKit.h>
#import "WaterLoginViewController.h"

@interface WaterProjectViewController : UITableViewController
<UITableViewDelegate> {
    NSArray *_projectList;
}

@property (nonatomic) int *rowIDNum;
@property (nonatomic, strong) NSString *rowIDToTransfer;

@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *projectSpinner;

@end
```

```
//  WaterProjectViewController.m

#import "WaterProjectViewController.h"
#import "WaterViewController.h"
#import <RestKit/RestKit.h>

#import "RKProjects.h"

#import "WaterFieldViewController.h"

@interface WaterProjectViewController ()

@end
```

```
@implementation WaterProjectViewController

@synthesize rowIDNum, rowIDToTransfer, projectSpinner;

- (id)initWithStyle:(UITableViewStyle)style
{
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

-(void)loadProjects{
    // Restkit GET Project list
    RKObjectMapping *mapping = [RKObjectMapping mappingForClass:[RKProjects
class]];
    [mapping addAttributeMappingsFromDictionary:@{
                                        @"project_current":    @"projectCurrent",
                                        @"name":               @"name",
                                        @"id":                 @"iD",
                                        @"projects.projtype":  @"projectType"
                                        }];

    RKResponseDescriptor *responseDescriptor = [RKResponseDescriptor
responseDescriptorWithMapping:mapping method:RKRequestMethodGET
pathPattern:nil keyPath:nil statusCodes:nil];

    // Place the userToken into the 'savedToken' NSString
    NSString *savedToken = [[NSUserDefaults standardUserDefaults]
stringForKey:@"tokenName"];

    // create our URL request
    // Place the savedToken into the URL
    NSString *URLString = [[NSString alloc]
initWithFormat:@"https://erams.com/api/%@/projects/", savedToken];
```

```
    // create our URL based on the string
    NSURL *baseURL = [NSURL URLWithString:URLString];

    // perform a request to the server
    NSURLRequest *request = [NSURLRequest requestWithURL:baseURL];
    RKObjectRequestOperation *operation = [[RKObjectRequestOperation alloc]
initWithRequest:request responseDescriptors:@[ responseDescriptor ]];
    [operation setCompletionBlockWithSuccess:^(RKObjectRequestOperation *operation,
RKMappingResult *result) {
        // we have successfully retrieved data to *result

        NSArray* projectList = [result array];
        NSLog(@"result array %@", projectList);
        // In RKProjects we return project save, put all names into NSArray projectList
        _projectList = projectList;

        // Once the projectVC is loaded, place project names into table
        if (self.isViewLoaded) {
            [projectSpinner stopAnimating];
            [self.tableView reloadData];
        }

    } failure:^(RKObjectRequestOperation *operation, NSError *error){
        RKLogError(@"My Operation failed with error: %@", error);
    }];
    [operation start];

}

- (void)viewDidLoad
{
    // Start loading indicator when view is visible
    [projectSpinner startAnimating];
    // Calling loadProjects method
    [self loadProjects];

    [super viewDidLoad];

    // Back button white text
    [[UINavigationBar appearance] setTintColor:[UIColor whiteColor]];

    // set navigation bar to black (top of view with buttons)
    self.navigationController.navigationBar.barStyle = UIBarStyleBlackOpaque;
}
```

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Only 1 section within the UITableView
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
    // Rows should equal the number of projects
    return [_projectList count];
}

- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"ProjectIdentifier";
    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];

    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
    }

    // List each object in array as a separate cell in the table
    RKProjects *projectArray = [_projectList objectAtIndex:indexPath.row];

    cell.textLabel.text = [projectArray name];

    NSArray *projType = [[NSArray alloc] init];
    projType = [projectArray projectType];

    if (![projType indexOfObject:@"irrigation"]){
        // If they are irrigation projects, add raindrop icon
        cell.imageView.image = [UIImage imageNamed:@"raindrop_uitablecell2.png"];
    }

    return cell;
}
```

```
#pragma mark
#pragma mark - Table view delegate
#pragma mark - Table view data source
// This declares the segue and maintains the title from the cell to the detailed view (gauge)
// Create the segue between the entire view and the detail view, then implement this code

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath
{
    //rowIDNum returns int which cell the user selected
    rowIDNum = indexPath.row;

    // Going to RKProjects to obtain the field ID at the same number that the user selected
    //(Matches cell number and ID number in the array
    RKProjects *iDNumber = [_projectList objectAtIndex:rowIDNum];

    //rowIDToTransfer gives the ID of the project selected
    self.rowIDToTransfer = [[NSString alloc] init];
    self.rowIDToTransfer = [iDNumber projectCurrent];
    NSLog(@"RowIDToTransfer %@", self.rowIDToTransfer);
    // save to NSUserDefaults
    [[NSUserDefaults standardUserDefaults] setObject:rowIDToTransfer
forKey:@"currentProjectID"];

    NSString *iD = [[NSString alloc] init];
    iD = [iDNumber iD];
    [[NSUserDefaults standardUserDefaults] setObject:iD forKey:@"ProjectID"];

    NSString *projectName = [[NSString alloc] init];
    projectName = [iDNumber name];
    [[NSUserDefaults standardUserDefaults] setObject:projectName
forKey:@"projectNameSaved"];

    // If the below text is deleted, a push segue from projects to fields needs to be created
    // Below sets the fields title to the project name
    UIBarButtonItem *backButton = [[UIBarButtonItem alloc] initWithTitle:@"Projects"
style:UIBarButtonItemStyleBordered target:nil action:nil];
    [self.navigationItem setBackBarButtonItem:backButton];

    UITableViewCell *selectedCell = [tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath];
    NSString *currentLabel = selectedCell.textLabel.text;

    WaterFieldViewController *fieldVC = [self.storyboard
instantiateViewControllerWithIdentifier:@"WaterFieldViewController"];
```

```
    // Title for next view will be: "FieldName Status"
    fieldVC.navigationItem.title = [NSString stringWithFormat:@"%@", currentLabel];

    [self.navigationController pushViewController:fieldVC animated:YES];
}

@end
```

```
//  RKProjects.h

#import <Foundation/Foundation.h>

@interface RKProjects : NSObject

@property (nonatomic, copy) NSString *projectCurrent;
@property (nonatomic, copy) NSString *sharable;
@property (nonatomic, copy) NSString *name;
@property (nonatomic, copy) NSString *createdBy;
@property (nonatomic, copy) NSString *placemarkLayerId;
@property (nonatomic, copy) NSString *iD;

@property (nonatomic, copy) NSArray *projectType;
@property (nonatomic, copy) NSString *currentEditingLayerID;
@property (nonatomic, copy) NSString *modelingArea;

@end
```

```
//  RKProjects.m

#import "RKProjects.h"

@implementation RKProjects

- (NSString *)description
{
    return [NSString stringWithFormat:@"%@ , %@", self.name, self.projectType];
}

@end
```

```
//  WaterFieldViewController.h
```

```
#import <UIKit/UIKit.h>
#import "WaterProjectViewController.h"

@interface WaterFieldViewController : WaterProjectViewController
<UITableViewDelegate>{
      NSArray *_fieldList;
}

@property (nonatomic) int fieldCellSelected;
@property (nonatomic) int rowSelected;
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *fieldSpinner;

@end
```

```
//  WaterFieldViewController.m

#import "WaterFieldViewController.h"
#import "WaterViewController.h"

#import <RestKit/RestKit.h>
#import "RKFields.h"

@interface WaterFieldViewController ()

@end

@implementation WaterFieldViewController

@synthesize fieldCellSelected, rowSelected, fieldSpinner;

- (id)initWithStyle:(UITableViewStyle)style
{
   self = [super initWithStyle:style];
   if (self) {
      // Custom initialization
   }
   return self;
}

-(void)loadFields{
   //Load project current (project id)
   NSString *projectIDNum = [[NSUserDefaults standardUserDefaults]
stringForKey:@"currentProjectID"];

   RKObjectMapping *mapping = [RKObjectMapping mappingForClass:[RKFields
```

```
class]];
    [mapping addAttributeMappingsFromDictionary:@{
                                        @"crop_type":           @"cropType",
                                        @"cutting_date_1":      @"cuttingDate1",
                                        @"cutting_date_2":      @"cuttingDate2",
                                        @"cutting_date_3":      @"cuttingDate3",
                                        @"cutting_date_4":      @"cuttingDate4",
                                        @"irrigation_method":   @"irrigationMethod",
                                        @"name":                @"fieldName",
                                        @"emergence_date":      @"emergenceDate",
                                        @"irrigation_type":     @"irrigationType",
                                        @"weather_stations":    @"weatherStations",
                                        @"irrigation_efficiency": @"irrigationEfficiency",
                                        @"planting_date":       @"plantingDate",
                                        @"gid":                 @"gid",

                                        @"mad":                 @"mad",
                                        @"gdd_base_temp":       @"gddBaseTemp",
                                        @"gdd_max_temp":        @"gddMaxTemp",
                                        @"control_depth":       @"controlDepth",
                                        @"init_soil_def":       @"initialSoilDef",
                                        @"irrigation_capacity":  @"irrigationCapacity",
                                        @"irrigation_frequency": @"irrigationFrequency",
                                        @"stage1_perc_growth":   @"stage1PercGrowth",
                                        @"stage2_perc_growth":   @"stage2PercGrowth",
                                        @"stage3_perc_growth":   @"stage3PercGrowth",
                                        @"stage1_crop_coeff":    @"stage1CropCoeff",
                                        @"stage2_crop_coeff":    @"stage2CropCoeff",
                                        @"stage3_crop_coeff":    @"stage3CropCoeff",
                                        @"harvest_date":        @"harvestDate",
                                        @"gdd_maturity":        @"gddMaturity"
                                        }];

    RKResponseDescriptor *responseDescriptor = [RKResponseDescriptor
responseDescriptorWithMapping:mapping method:RKRequestMethodGET
pathPattern:nil keyPath:nil statusCodes:nil];
    //get saved user token
    NSString *savedToken = [[NSUserDefaults standardUserDefaults]
stringForKey:@"tokenName"];

    // create our URL request
    NSString *URLString = [[NSString alloc]
initWithFormat:@"https://erams.com/api/%@/irrigation/cropfields/?proj=%@",
savedToken, projectIDNum];

    // create our URL based on the string
```

```
    NSURL *baseURL = [NSURL URLWithString:URLString];

    NSLog(@"Field List URL: %@", baseURL);

    // perform a GET request to the server
    NSURLRequest *request = [NSURLRequest requestWithURL:baseURL];
    RKObjectRequestOperation *operation = [[RKObjectRequestOperation alloc]
initWithRequest:request responseDescriptors:@[ responseDescriptor ]];
    [operation setCompletionBlockWithSuccess:^(RKObjectRequestOperation *operation,
RKMappingResult *result) {
        // we have successfully retrieved data to *result
        NSArray* fieldList = [result array];
        NSLog(@"field result array %@", fieldList);

        // Shows alert if there are no fields in the array
        // This may happen if a user creates a project but doesn't create fields within the
project
        if ([result.array count] == 0) {
            UIAlertView *noFields = [[UIAlertView alloc] initWithTitle: @"No Fields"
                                                               message:@"There are no fields within this
project."
                                                              delegate:self
                                                     cancelButtonTitle:@"OK"
                                                     otherButtonTitles:nil];
            [noFields show];
            [fieldSpinner stopAnimating];
        }

        _fieldList = fieldList;

        if (self.isViewLoaded) {
            [fieldSpinner stopAnimating];
            [self.tableView reloadData];
        }

    } failure:^(RKObjectRequestOperation *operation, NSError *error){
        RKLogError(@"My Operation failed with error: %@", error);

        //Shows alert if there is an error when with get response. Usually no fields are in the
project.
        UIAlertView *errorAlert = [[UIAlertView alloc] initWithTitle: @"Error"
                                                             message:@"Please make sure that there are fields
within this project."
                                                            delegate:self
                                                   cancelButtonTitle:@"OK"
                                                   otherButtonTitles:nil];
```

```
        [errorAlert show];
        [fieldSpinner stopAnimating];

    }];
    [operation start];
}

- (void)viewDidLoad
{
    [fieldSpinner startAnimating];
    [self loadFields];
    [super viewDidLoad];

    //Back & menu buttons gold color
    self.navigationController.navigationBar.tintColor = [UIColor colorWithRed:.72
green:.55 blue:.15 alpha:1];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    NSLog(@"Number of Fields %lu", (unsigned long)_fieldList.count);
    return _fieldList.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"FieldIdentifier";
    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:@"FieldIdentifier"
```

```
                                    forIndexPath:indexPath];

    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
    }

    // Configure the cell...
    RKFields *field = [_fieldList objectAtIndex:indexPath.row];
    cell.textLabel.text = [field fieldName];

    return cell;
}


- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath
{
    rowSelected = indexPath.row;

    // Within this method, we need to save crop attributes to NSUserDefaults for the
specific field selected
        // Going to RKFields to obtain attributes at the same number that the user selected
    RKFields *objectNumberSelected = [_fieldList objectAtIndex:rowSelected];


    NSString *quote = @"\"";

    // Cutting dates:
    NSString *cuttingDate1Saved = [[NSString alloc] init];
    cuttingDate1Saved = [objectNumberSelected cuttingDate1];
    [[NSUserDefaults standardUserDefaults] setObject:cuttingDate1Saved
forKey:@"cuttingDate1Saved"];

    NSLog(@"Original Cutting date: %@", cuttingDate1Saved);

    if ([cuttingDate1Saved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        cuttingDate1Saved = [quote stringByAppendingString:cuttingDate1Saved];
        cuttingDate1Saved = [cuttingDate1Saved stringByAppendingString:quote];
        [[NSUserDefaults standardUserDefaults] setObject:cuttingDate1Saved
forKey:@"cuttingDate1Saved"];
        NSLog(@"Cutting 1 is %@", cuttingDate1Saved);

    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
```

```
forKey:@"cuttingDate1Saved"];
        NSLog(@"No Cutting 1 Date");
    }

    NSString *cuttingDate2Saved = [[NSString alloc] init];
    cuttingDate2Saved = [objectNumberSelected cuttingDate2];
    [[NSUserDefaults standardUserDefaults] setObject:cuttingDate2Saved
forKey:@"cuttingDate2Saved"];

    if ([cuttingDate2Saved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        cuttingDate2Saved = [quote stringByAppendingString:cuttingDate2Saved];
        cuttingDate2Saved = [cuttingDate2Saved stringByAppendingString:quote];
        [[NSUserDefaults standardUserDefaults] setObject:cuttingDate2Saved
forKey:@"cuttingDate2Saved"];
        NSLog(@"Cutting 2 is %@", cuttingDate2Saved);

    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"cuttingDate2Saved"];
        NSLog(@"No Cutting 2 Date");
    }

    NSString *cuttingDate3Saved = [[NSString alloc] init];
    cuttingDate3Saved = [objectNumberSelected cuttingDate3];
    [[NSUserDefaults standardUserDefaults] setObject:cuttingDate3Saved
forKey:@"cuttingDate3Saved"];

    if ([cuttingDate3Saved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        cuttingDate3Saved = [quote stringByAppendingString:cuttingDate3Saved];
        cuttingDate3Saved = [cuttingDate3Saved stringByAppendingString:quote];
        [[NSUserDefaults standardUserDefaults] setObject:cuttingDate3Saved
forKey:@"cuttingDate3Saved"];
        NSLog(@"Cutting 3 is %@", cuttingDate3Saved);

    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"cuttingDate3Saved"];
        NSLog(@"No Cutting 3 Date");
    }

    NSString *cuttingDate4Saved = [[NSString alloc] init];
    cuttingDate4Saved = [objectNumberSelected cuttingDate4];
    [[NSUserDefaults standardUserDefaults] setObject:cuttingDate4Saved
forKey:@"cuttingDate4Saved"];
```

```
    if ([cuttingDate4Saved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        cuttingDate4Saved = [quote stringByAppendingString:cuttingDate4Saved];
        cuttingDate4Saved = [cuttingDate4Saved stringByAppendingString:quote];
        [[NSUserDefaults standardUserDefaults] setObject:cuttingDate4Saved
forKey:@"cuttingDate4Saved"];
        NSLog(@"Cutting 4 is %@", cuttingDate4Saved);

    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"cuttingDate4Saved"];
        NSLog(@"No Cutting 4 Date");
    }

    //Saving crop type
    NSString *cropTypeSaved = [[NSString alloc] init];
    cropTypeSaved = [objectNumberSelected cropType];
    NSLog(@"Crop type: %@", cropTypeSaved);
    [[NSUserDefaults standardUserDefaults] setObject:cropTypeSaved
forKey:@"cropTypeSaved"];

    //Saving irrigation method
    NSString *irrigationMethodSaved = [[NSString alloc] init];
    irrigationMethodSaved = [objectNumberSelected irrigationMethod];
    NSLog(@"irrigation method: %@", irrigationMethodSaved);
    [[NSUserDefaults standardUserDefaults] setObject:irrigationMethodSaved
forKey:@"irrigationMethodSaved"];

    //Saving field name
    NSString *fieldNameSaved = [[NSString alloc] init];
    fieldNameSaved = [objectNumberSelected fieldName];
    NSLog(@"field Name: %@", fieldNameSaved);
    [[NSUserDefaults standardUserDefaults] setObject:fieldNameSaved
forKey:@"fieldNameSaved"];

    //Saving emergence Date
    NSString *emergenceDateSaved = [[NSString alloc] init];
    emergenceDateSaved = [objectNumberSelected emergenceDate];
    NSLog(@"emergence date: %@", emergenceDateSaved);
    [[NSUserDefaults standardUserDefaults] setObject:emergenceDateSaved
forKey:@"emergenceDateSaved"];

    //Saving irrigation type
    NSString *irrigationTypeSaved = [[NSString alloc] init];
    irrigationTypeSaved = [objectNumberSelected irrigationType];
```

```
    NSLog(@"irrigation type: %@", irrigationTypeSaved);
    [[NSUserDefaults standardUserDefaults] setObject:irrigationTypeSaved
forKey:@"irrigationTypeSaved"];

    // Saving weather stations
    NSString *weatherStaions = [[NSString alloc] init];
    weatherStaions = [objectNumberSelected weatherStations];
    NSLog(@"Weather Staions: %@", weatherStaions);
    [[NSUserDefaults standardUserDefaults] setObject:weatherStaions
forKey:@"weatherStationsSaved"];

    //Saving irrigation efficiency
    NSString *irrigationEfficiencySaved = [[NSString alloc] init];
    irrigationEfficiencySaved = [objectNumberSelected irrigationEfficiency];
    NSLog(@"irrigation efficiency: %@", irrigationEfficiencySaved);
    [[NSUserDefaults standardUserDefaults] setObject:irrigationEfficiencySaved
forKey:@"irrigationEfficiencySaved"];

    //Saving planting date
    NSString *plantingDateSaved = [[NSString alloc] init];
    plantingDateSaved = [objectNumberSelected plantingDate];
    NSLog(@"planting date: %@", plantingDateSaved);
    [[NSUserDefaults standardUserDefaults] setObject:plantingDateSaved
forKey:@"plantingDateSaved"];

    // Saving GID
    NSString *gidSaved = [[NSString alloc] init];
    gidSaved = [objectNumberSelected gid];
    NSLog(@"gid: %@", gidSaved);
    [[NSUserDefaults standardUserDefaults] setObject:gidSaved forKey:@"gidSaved"];

    //Saving MAD
    NSString *madSaved = [[NSString alloc] init];
    madSaved = [objectNumberSelected mad];
    [[NSUserDefaults standardUserDefaults] setObject:madSaved forKey:@"madSaved"];
    if ([madSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:madSaved
forKey:@"madSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null" forKey:@"madSaved"];
    }

    //Saving gdd base temp
    NSString *gddBaseTempSaved = [[NSString alloc] init];
    gddBaseTempSaved = [objectNumberSelected gddBaseTemp];
```

```
    [[NSUserDefaults standardUserDefaults] setObject:gddBaseTempSaved
forKey:@"gddBaseTempSaved"];
    if ([gddBaseTempSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:gddBaseTempSaved
forKey:@"gddBaseTempSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"gddBaseTempSaved"];
    }

    //Saving gdd max temp
    NSString *gddMaxTempSaved = [[NSString alloc] init];
    gddMaxTempSaved = [objectNumberSelected gddMaxTemp];
    [[NSUserDefaults standardUserDefaults] setObject:gddMaxTempSaved
forKey:@"gddMaxTempSaved"];
    if ([gddMaxTempSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:gddMaxTempSaved
forKey:@"gddMaxTempSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"gddMaxTempSaved"];
    }

    // Saving control depth
    NSString *controlDepthSaved = [[NSString alloc] init];
    controlDepthSaved = [objectNumberSelected controlDepth];
    [[NSUserDefaults standardUserDefaults] setObject:controlDepthSaved
forKey:@"controlDepthSaved"];
    if ([controlDepthSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:controlDepthSaved
forKey:@"controlDepthSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"controlDepthSaved"];
    }

    // Saving initial Soil deficit
    NSString *initialSoilDefictSaved = [[NSString alloc] init];
    initialSoilDefictSaved = [objectNumberSelected initialSoilDef];
    [[NSUserDefaults standardUserDefaults] setObject:initialSoilDefictSaved
forKey:@"initialSoilDefictSaved"];
    if ([initialSoilDefictSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
```

```
        [[NSUserDefaults standardUserDefaults] setObject:initialSoilDeficitSaved
forKey:@"initialSoilDeficitSaved"];
        NSLog(@"Initial Soil Def is %@", initialSoilDeficitSaved);
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"initialSoilDeficitSaved"];
    }

    // Saving irrigation capacity
    NSString *irrigationCapacitySaved = [[NSString alloc] init];
    irrigationCapacitySaved = [objectNumberSelected irrigationCapacity];
    [[NSUserDefaults standardUserDefaults] setObject:irrigationCapacitySaved
forKey:@"irrigationCapacitySaved"];
    if ([irrigationCapacitySaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:irrigationCapacitySaved
forKey:@"irrigationCapacitySaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"irrigationCapacitySaved"];
    }

    // Saving irrigation frequency
    NSString *irrigationFrequencySaved = [[NSString alloc] init];
    irrigationEfficiencySaved = [objectNumberSelected irrigationFrequency];
    [[NSUserDefaults standardUserDefaults] setObject:irrigationFrequencySaved
forKey:@"irrigationFrequencySaved"];
    if ([irrigationFrequencySaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:irrigationFrequencySaved
forKey:@"irrigationFrequencySaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"irrigationFrequencySaved"];
    }

    // Saving stage 1 per growth
    NSString *stage1PercGrowthSaved = [[NSString alloc] init];
    stage1PercGrowthSaved = [objectNumberSelected stage1PercGrowth];
    [[NSUserDefaults standardUserDefaults] setObject:stage1PercGrowthSaved
forKey:@"stage1PercGrowthSaved"];
    if ([stage1PercGrowthSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:stage1PercGrowthSaved
forKey:@"stage1PercGrowthSaved"];
    } else {
```

```
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"stage1PercGrowthSaved"];
    }

    // Saving stage 2 per growth
    NSString *stage2PercGrowthSaved = [[NSString alloc] init];
    stage2PercGrowthSaved = [objectNumberSelected stage2PercGrowth];
    [[NSUserDefaults standardUserDefaults] setObject:stage2PercGrowthSaved
forKey:@"stage2PercGrowthSaved"];
    if ([stage2PercGrowthSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:stage2PercGrowthSaved
forKey:@"stage2PercGrowthSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"stage2PercGrowthSaved"];
    }

    // Saving stage 3 per growth
    NSString *stage3PercGrowthSaved = [[NSString alloc] init];
    stage3PercGrowthSaved = [objectNumberSelected stage3PercGrowth];
    [[NSUserDefaults standardUserDefaults] setObject:stage3PercGrowthSaved
forKey:@"stage3PercGrowthSaved"];
    if ([stage3PercGrowthSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:stage3PercGrowthSaved
forKey:@"stage3PercGrowthSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"stage3PercGrowthSaved"];
    }

    // Saving stage 1 crop coeff
    NSString *stage1CropCoeffSaved = [[NSString alloc] init];
    stage1CropCoeffSaved = [objectNumberSelected stage1CropCoeff];
    [[NSUserDefaults standardUserDefaults] setObject:stage1CropCoeffSaved
forKey:@"stage1CropCoeffSaved"];
    if ([stage1CropCoeffSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:stage1CropCoeffSaved
forKey:@"stage1CropCoeffSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"stage1CropCoeffSaved"];
    }
```

```
    //Saving stage 2 crop coeff
    NSString *stage2CropCoeffSaved = [[NSString alloc] init];
    stage2CropCoeffSaved = [objectNumberSelected stage2CropCoeff];
    [[NSUserDefaults standardUserDefaults] setObject:stage2CropCoeffSaved
forKey:@"stage2CropCoeffSaved"];
    if ([stage2CropCoeffSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:stage2CropCoeffSaved
forKey:@"stage2CropCoeffSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"stage2CropCoeffSaved"];
    }

    //Saving stage 3 crop coeff
    NSString *stage3CropCoeffSaved = [[NSString alloc] init];
    stage3CropCoeffSaved = [objectNumberSelected stage3CropCoeff];
    [[NSUserDefaults standardUserDefaults] setObject:stage3CropCoeffSaved
forKey:@"stage3CropCoeffSaved"];
    if ([stage3CropCoeffSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:stage3CropCoeffSaved
forKey:@"stage3CropCoeffSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"stage3CropCoeffSaved"];
    }

    // saving harvest date
    NSString *harvestDateSaved = [[NSString alloc] init];
    harvestDateSaved = [objectNumberSelected harvestDate];
    [[NSUserDefaults standardUserDefaults] setObject:harvestDateSaved
forKey:@"harvestDateSaved"];
    if ([harvestDateSaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        harvestDateSaved = [quote stringByAppendingString:harvestDateSaved];
        harvestDateSaved = [harvestDateSaved stringByAppendingString:quote];
        [[NSUserDefaults standardUserDefaults] setObject:harvestDateSaved
forKey:@"harvestDateSaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"harvestDateSaved"];
    }

    // saving gdd maturity
    NSString *gddMaturitySaved = [[NSString alloc] init];
```

```
    gddMaturitySaved = [objectNumberSelected gddMaturity];
    [[NSUserDefaults standardUserDefaults] setObject:gddMaturitySaved
forKey:@"gddMaturitySaved"];
    if ([gddMaturitySaved rangeOfString:@"null"].location == NSNotFound){
        //string does not contain null
        [[NSUserDefaults standardUserDefaults] setObject:gddMaturitySaved
forKey:@"gddMaturitySaved"];
    } else {
        [[NSUserDefaults standardUserDefaults] setObject:@"null"
forKey:@"gddMaturitySaved"];
    }

    //Set the back button to fields
    UIBarButtonItem *backButton = [[UIBarButtonItem alloc] initWithTitle:@"Fields"
style:UIBarButtonItemStyleBordered target:nil action:nil];
    [self.navigationItem setBackBarButtonItem:backButton];

    fieldCellSelected = indexPath.row;
    NSLog(@"User Selected: %d", fieldCellSelected);

    UITableViewCell *selectedCell = [tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath];
    NSString *currentLabel = selectedCell.textLabel.text;

    WaterViewController *gaugeVC = [self.storyboard
instantiateViewControllerWithIdentifier:@"WaterViewController"];

    // Title for next view will be: "FieldName Status"
    gaugeVC.navigationItem.title = [NSString stringWithFormat:@"%@ Status",
currentLabel];

    [self.navigationController pushViewController:gaugeVC animated:YES];
}

@end
```

```
//  RKFields.h

#import <Foundation/Foundation.h>

@interface RKFields : NSObject

@property (nonatomic, copy) NSString *fieldName;
@property (nonatomic, copy) NSString *cropType;
@property (nonatomic, copy) NSString *cuttingDate1;
```

```
@property (nonatomic, copy) NSString *cuttingDate2;
@property (nonatomic, copy) NSString *cuttingDate3;
@property (nonatomic, copy) NSString *cuttingDate4;
@property (nonatomic, copy) NSString *plantingDate;
@property (nonatomic, copy) NSString *emergenceDate;
@property (nonatomic, copy) NSString *irrigationFrequency;
@property (nonatomic, copy) NSString *irrigationType;
@property (nonatomic, copy) NSString *irrigationMethod;
@property (nonatomic, copy) NSString *weatherStations;
@property (nonatomic, copy) NSString *irrigationEfficiency;
@property (nonatomic, copy) NSString *gid;
@property (nonatomic, copy) NSString *mad;

@property (nonatomic, copy) NSString *gddBaseTemp;
@property (nonatomic, copy) NSString *gddMaxTemp;
@property (nonatomic, copy) NSString *controlDepth;
@property (nonatomic, copy) NSString *initialSoilDef;
@property (nonatomic, copy) NSString *irrigationCapacity;
@property (nonatomic, copy) NSString *stage1PercGrowth;
@property (nonatomic, copy) NSString *stage2PercGrowth;
@property (nonatomic, copy) NSString *stage3PercGrowth;
@property (nonatomic, copy) NSString *stage1CropCoeff;
@property (nonatomic, copy) NSString *stage2CropCoeff;
@property (nonatomic, copy) NSString *stage3CropCoeff;
@property (nonatomic, copy) NSString *harvestDate;
@property (nonatomic, copy) NSString *gddMaturity;

@end
```

```
//  RKFields.m

#import "RKFields.h"

@implementation RKFields

- (NSString *)description
{
    return [NSString stringWithFormat:@"%@ : %@ ", self.fieldName, self.cropType];
}

@end
```

```
//  RKFields.m
```

```
#import "RKFields.h"

@implementation RKFields

- (NSString *)description
{
    return [NSString stringWithFormat:@"%@ : %@ ", self.fieldName, self.cropType];
}

@end
```

```
//  WaterViewController.m

#import "WaterViewController.h"
#import "WaterGauge.h"
#import "WaterWeatheriPadView.h"
#import <QuartzCore/QuartzCore.h>
#import <RestKit/RestKit.h>
#import "RKAttributes.h"
#import "RKAddIrrigation.h"
#import "RKAddIrrResponse.h"

@interface WaterViewController ()
// Create NSString properties to handle location in array for the current and prev. date.
@property (nonatomic, strong) NSString *indexCurrentDate;
@property (nonatomic, strong) NSString *indexWeatherPrevDate;

@property (nonatomic, strong) NSString *percGrowth;

@end

@implementation WaterViewController

@synthesize fCLabel, currentLevelLabel, wPLabel, mADLabel, daysRemainingLabel,
growthStageLabel, addButton, weatherButton, gaugeSpinner, indexCurrentDate,
indexWeatherPrevDate, gaugeView, weatheriPadView, percGrowth;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
```

```
-(void)loadFieldAttributes{

    // Retrieve saved values from NSUserDefaults for URL link
    NSString *savedToken = [[NSUserDefaults standardUserDefaults]
stringForKey:@"tokenName"];
    NSString *currentProjectIDNum = [[NSUserDefaults standardUserDefaults]
stringForKey:@"currentProjectID"];
    NSString *gid = [[NSUserDefaults standardUserDefaults]
stringForKey:@"gidSaved"];
    NSString *cropType = [[NSUserDefaults standardUserDefaults]
stringForKey:@"cropTypeSaved"];
    NSString *irrigationMethod = [[NSUserDefaults standardUserDefaults]
stringForKey:@"irrigationMethodSaved"];
    NSString *fieldName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"fieldNameSaved"];
    NSString *emergenceDate = [[NSUserDefaults standardUserDefaults]
stringForKey:@"emergenceDateSaved"];
    NSString *irrigationType = [[NSUserDefaults standardUserDefaults]
stringForKey:@"irrigationTypeSaved"];
    NSString *weatherStations = [[NSUserDefaults standardUserDefaults]
stringForKey:@"weatherStationsSaved"];
    NSString *irrigationEfficiency = [[NSUserDefaults standardUserDefaults]
stringForKey:@"irrigationEfficiencySaved"];
    NSString *plantingDate = [[NSUserDefaults standardUserDefaults]
stringForKey:@"plantingDateSaved"];
    NSString *cuttingDate1 = [[NSUserDefaults standardUserDefaults]
stringForKey:@"cuttingDate1Saved"];
    NSString *cuttingDate2 = [[NSUserDefaults standardUserDefaults]
stringForKey:@"cuttingDate2Saved"];
    NSString *cuttingDate3 = [[NSUserDefaults standardUserDefaults]
stringForKey:@"cuttingDate3Saved"];
    NSString *cuttingDate4 = [[NSUserDefaults standardUserDefaults]
stringForKey:@"cuttingDate4Saved"];

    NSString *mad = [[NSUserDefaults standardUserDefaults]
stringForKey:@"madSaved"];
    NSString *gddBaseTemp = [[NSUserDefaults standardUserDefaults]
stringForKey:@"gddBaseTempSaved"];
    NSString *gddMaxTemp = [[NSUserDefaults standardUserDefaults]
stringForKey:@"gddMaxTempSaved"];
    NSString *controlDepth = [[NSUserDefaults standardUserDefaults]
stringForKey:@"controlDepthSaved"];
    NSString *initialSoilDef = [[NSUserDefaults standardUserDefaults]
stringForKey:@"initialSoilDefictSaved"];
    NSString *irrigationCapacity = [[NSUserDefaults standardUserDefaults]
```

```
stringForKey:@"irrigationCapacitySaved"];
    NSString *stage1PercGrowth = [[NSUserDefaults standardUserDefaults]
stringForKey:@"stage1PercGrowthSaved"];
    NSString *stage2PercGrowth = [[NSUserDefaults standardUserDefaults]
stringForKey:@"stage2PercGrowthSaved"];
    NSString *stage3PercGrowth = [[NSUserDefaults standardUserDefaults]
stringForKey:@"stage3PercGrowthSaved"];
    NSString *stage1CropCoeff = [[NSUserDefaults standardUserDefaults]
stringForKey:@"stage1CropCoeffSaved"];
    NSString *stage2CropCoeff = [[NSUserDefaults standardUserDefaults]
stringForKey:@"stage2CropCoeffSaved"];
    NSString *stage3CropCoeff = [[NSUserDefaults standardUserDefaults]
stringForKey:@"stage3CropCoeffSaved"];
    NSString *harvestDate = [[NSUserDefaults standardUserDefaults]
stringForKey:@"harvestDateSaved"];
    NSString *gddMaturity = [[NSUserDefaults standardUserDefaults]
stringForKey:@"gddMaturitySaved"];

    NSLog(@"check soil def: %@", initialSoilDef);

    RKObjectMapping *mapping = [RKObjectMapping mappingForClass:[RKAttributes
class]];
    [mapping addAttributeMappingsFromDictionary:@{
                                        @"awc":         @"awc",
                                        @"fc":          @"fc",
                                        @"pwp":         @"pwp",
                                        @"paw":         @"paw",
                                        @"paw_at_mad":  @"pawAtMad",
                                        @"date":        @"dateValue",
                                        @"et_c":        @"etc",
                                        @"gdd":         @"gdd",
                                        @"def_cur":     @"deficitCurrent",
                                        @"et_r":        @"etr",
                                        @"perc_growth": @"percentGrowth",
                                        @"depth_rz":    @"depthRootZone",
                                        @"precip":      @"precipitation",
                                        @"irrigation":  @"irrigationFromWeb",
                                        @"k_cr":        @"kcr",
                                        @"t_max_temp":  @"tempMax",
                                        @"k_s":         @"ks",
                                        @"def_mad":     @"defMad",
                                        @"t_min_temp":  @"tempMin"
                                        }];

    // Register our mappings with the provider using a response descriptor
    RKResponseDescriptor *responseDescriptor = [RKResponseDescriptor
```

```
responseDescriptorWithMapping:mapping method:RKRequestMethodGET
pathPattern:nil keyPath:nil statusCodes:nil];

    // create our URL request
    NSString *URLString = [[NSString alloc]
initWithFormat:@"https://erams.com/api/%@/irrigation/cropschedule/?field_attrs={\"id\
":%@,\"gid\":%@,\"name\":\"%@\",\"crop_type\":\"%@\",\"planting_date\":\"%@\",\"e
mergence_date\":\"%@\",\"weather_stations\":\"%@\",\"irrigation_type\":\"%@\",\"irriga
tion_method\":\"%@\",\"irrigation_efficiency\":%@,\"cutting_date_1\":%@,\"cutting_da
te_2\":%@,\"cutting_date_3\":%@,\"cutting_date_4\":%@,\"mad\":%@,\"gdd_base_tem
p\":%@,\"gdd_max_temp\":%@,\"control_depth\":%@,\"init_soil_def\":%@,\"irrigation
_capacity\":%@,\"stage1_perc_growth\":%@,\"stage2_perc_growth\":%@,\"stage3_perc
_growth\":%@,\"stage1_crop_coeff\":%@,\"stage2_crop_coeff\":%@,\"stage3_crop_coe
ff\":%@,\"harvest_date\":%@,\"gdd_maturity\":%@}&proj=%@", savedToken, gid, gid,
fieldName, cropType, plantingDate, emergenceDate, weatherStations, irrigationType,
irrigationMethod, irrigationEfficiency, cuttingDate1, cuttingDate2, cuttingDate3,
cuttingDate4, mad, gddBaseTemp, gddMaxTemp, controlDepth, initialSoilDef,
irrigationCapacity, /*irrigationFrequency,*/ stage1PercGrowth, stage2PercGrowth,
stage3PercGrowth, stage1CropCoeff, stage2CropCoeff, stage3CropCoeff, harvestDate,
gddMaturity, currentProjectIDNum];

    // Fix the encoding issue - URL will replace " with %22
    // Please note that within the URL \" means to add the (") without them meaning its the
end of the string
    NSString *strURL = [URLString
stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];

    // create our URL based on the string
    NSURL *baseURL = [NSURL URLWithString:strURL];
    NSLog(@"Field Attribute URL: %@", strURL);

    // perform a request to the server
    NSURLRequest *request = [NSURLRequest requestWithURL:baseURL];
    RKObjectRequestOperation *operation = [[RKObjectRequestOperation alloc]
initWithRequest:request responseDescriptors:@[ responseDescriptor ]];
    [operation setCompletionBlockWithSuccess:^(RKObjectRequestOperation *operation,
RKMappingResult *result) {
    // we have successfully retrieved data to *result

    NSArray* fieldAttributes = [result array];
    _fieldAttributes = fieldAttributes;

        // If the view is loaded, update the profile gauge values and redraw
        if (self.isViewLoaded) {
            NSLog(@"ETc Vals: %@", _fieldAttributes);
            //reload bucket gauge
```

```
            [self updateBucketGauge];
            [gaugeSpinner stopAnimating];

            //Allows user to press buttons now that the view is loaded
            self.view.userInteractionEnabled = YES;
        }

    } failure:^(RKObjectRequestOperation *operation, NSError *error){
    RKLogError(@"My Operation failed with error: %@", error);

        //Shows alert if there is not any crop attributes within the field.
        UIAlertView *noFieldAttributes = [[UIAlertView alloc] initWithTitle: @"Field
Attributes"
                                                                    message:@"Please set up field attributes online
before continuing."
                                                                   delegate:self
                                                          cancelButtonTitle:@"OK"
                                                          otherButtonTitles:nil];
        [noFieldAttributes show];
        [gaugeSpinner stopAnimating];

        //Allows user to press buttons now that the view is loaded
        self.view.userInteractionEnabled = YES;
    }];
    [operation start];
}

-(void)updateBucketGauge
{
    // We've successfully retrieved profile data, now lets sort through the array for:
    // Todays data (yesterdays weather) or last object in the case that the growing season is
over
    NSDate *todaysDate = [NSDate date];
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    [formatter setDateFormat:@"YYYY-MM-dd"];
    NSString *dateString = [formatter stringFromDate:todaysDate];

    // Search through each object in array
    for (int i = 0; i < [_fieldAttributes count]; i++) {
        NSString *indexValue = [[_fieldAttributes objectAtIndex:i] dateValue];
        NSLog(@"value: %@, index: %d", indexValue, i);
        // If indexValue string is equal to the current date, set the number in array to this
string
        if ([indexValue  isEqualToString:dateString]) {
            indexCurrentDate = [NSString stringWithFormat:@"%d", i];
            int iminus1 = i - 1;
```

```
            // For weather, we want yesterdays values so i-1 is set to indexWeatherPrevDate.
            indexWeatherPrevDate = [NSString stringWithFormat:@"%d", iminus1];
            // We found our values, get out of the loop.
            break;
        }
    }

    RKAttributes *objectNumberSelected = [_fieldAttributes
objectAtIndex:[indexCurrentDate intValue]];
    RKAttributes *weatherObjectNumberSelected = [_fieldAttributes
objectAtIndex:[indexWeatherPrevDate intValue]];

    if (indexCurrentDate == NULL) {
        // Current date was NULL, meaning we are outside of the growing season
        if (indexCurrentDate == NULL) {
            //If current date is not within growing season, choose the last object in array.
            objectNumberSelected = [_fieldAttributes lastObject];
            NSLog(@"Last item was selected: %@", objectNumberSelected);
            weatherObjectNumberSelected = [_fieldAttributes lastObject];
        }
    }

    NSLog(@"Final value for objectNumberSelected: %@", objectNumberSelected);

    // Save values needed for gauge from the desired location in array (date)

    NSString *awcSaved = [[NSString alloc] init];
    awcSaved = [objectNumberSelected awc];
    NSLog(@"awc: %@", awcSaved);
    [[NSUserDefaults standardUserDefaults] setObject:awcSaved forKey:@"awcSaved"];

    NSString *fcSaved = [[NSString alloc] init];
    fcSaved = [objectNumberSelected fc];
    NSLog(@"fc: %@", fcSaved);
    [[NSUserDefaults standardUserDefaults] setObject:fcSaved forKey:@"fcSaved"];

    NSString *pwpSaved = [[NSString alloc] init];
    pwpSaved = [objectNumberSelected pwp];
    NSLog(@"pwp: %@", pwpSaved);
    [[NSUserDefaults standardUserDefaults] setObject:pwpSaved forKey:@"pwpSaved"];

    NSString *pawSaved = [[NSString alloc] init];
    pawSaved = [objectNumberSelected paw];
    NSLog(@"paw: %@", pawSaved);
    [[NSUserDefaults standardUserDefaults] setObject:pawSaved forKey:@"pawSaved"];
```

```
    NSString *pawAtMadSaved = [[NSString alloc] init];
    pawAtMadSaved = [objectNumberSelected pawAtMad];
    NSLog(@"pawAtMad: %@", pawAtMadSaved);
    [[NSUserDefaults standardUserDefaults] setObject:pawAtMadSaved
forKey:@"pawAtMadSaved"];

    NSString *dateSaved = [[NSString alloc] init];
    dateSaved = [objectNumberSelected dateValue];
    NSLog(@"date: %@", dateSaved);
    [[NSUserDefaults standardUserDefaults] setObject:dateSaved forKey:@"dateSaved"];

    NSString *deficitCurrentSaved = [[NSString alloc] init];
    deficitCurrentSaved = [objectNumberSelected deficitCurrent];
    NSLog(@"deficitCurrent: %@", deficitCurrentSaved);
    [[NSUserDefaults standardUserDefaults] setObject:deficitCurrentSaved
forKey:@"deficitCurrentSaved"];

    NSString *defMADSaved = [[NSString alloc] init];
    defMADSaved = [objectNumberSelected defMad];
    NSLog(@"defMAD: %@", defMADSaved);
    [[NSUserDefaults standardUserDefaults] setObject:defMADSaved
forKey:@"defMADSaved"];

    //Add saved weather parameters here

    NSString *dateWeatherSaved = [[NSString alloc] init];
    dateWeatherSaved = [weatherObjectNumberSelected dateValue];
    NSLog(@"weather date: %@", dateWeatherSaved);
    [[NSUserDefaults standardUserDefaults] setObject:dateWeatherSaved
forKey:@"dateWeatherSaved"];

    NSString *etcSaved = [[NSString alloc] init];
    etcSaved = [weatherObjectNumberSelected etc];
    NSLog(@"etc: %@", etcSaved);
    [[NSUserDefaults standardUserDefaults] setObject:etcSaved forKey:@"etcSaved"];

    NSString *gddSaved = [[NSString alloc] init];
    gddSaved = [weatherObjectNumberSelected gdd];
    NSLog(@"gdd: %@", gddSaved);
    [[NSUserDefaults standardUserDefaults] setObject:gddSaved forKey:@"gddSaved"];

    NSString *etrSaved = [[NSString alloc] init];
    etrSaved = [weatherObjectNumberSelected etr];
    NSLog(@"etr: %@", etrSaved);
    [[NSUserDefaults standardUserDefaults] setObject:etrSaved forKey:@"etrSaved"];
```

```
    NSString *precipitationSaved = [[NSString alloc] init];
    precipitationSaved = [weatherObjectNumberSelected precipitation];
    NSLog(@"precip: %@", precipitationSaved);
    [[NSUserDefaults standardUserDefaults] setObject:precipitationSaved
forKey:@"precipitationSaved"];

    NSString *tempMaxSaved = [[NSString alloc] init];
    tempMaxSaved = [weatherObjectNumberSelected tempMax];
    NSLog(@"Max Temp: %@", tempMaxSaved);
    [[NSUserDefaults standardUserDefaults] setObject:tempMaxSaved
forKey:@"tempMaxSaved"];

    NSString *tempMinSaved = [[NSString alloc] init];
    tempMinSaved = [weatherObjectNumberSelected tempMin];
    NSLog(@"Min Temp: %@", tempMinSaved);
    [[NSUserDefaults standardUserDefaults] setObject:tempMinSaved
forKey:@"tempMinSaved"];

    NSString *percGrowthSaved = [[NSString alloc] init];
    percGrowthSaved = [weatherObjectNumberSelected percentGrowth];
    NSLog(@"Perc. Growth: %@", percGrowthSaved);
    [[NSUserDefaults standardUserDefaults] setObject:percGrowthSaved
forKey:@"percGrowthSaved"];

    NSString *ksSaved = [[NSString alloc] init];
    ksSaved = [weatherObjectNumberSelected ks];
    NSLog(@"ks: %@", ksSaved);
    [[NSUserDefaults standardUserDefaults] setObject:ksSaved forKey:@"ksSaved"];

    // Calling updateGaugeValues method
    [self updateGaugeValues];
}

- (void)viewDidLoad
{
    // Curve button edges
    addButton.layer.cornerRadius = 5;
    addButton.clipsToBounds = YES;
    weatherButton.layer.cornerRadius = 5;
    weatherButton.clipsToBounds = YES;

    // start loading indicator, call loadFieldAttributes method.
    [gaugeSpinner startAnimating];

    // Prohibits user from pressing buttons until the view is loaded
    self.view.userInteractionEnabled = NO;
```

```
    [self loadFieldAttributes];

    [super viewDidLoad];

}

-(void)updateGaugeValues
{
    // Call these methods to calculate various irrigation properties
    [self availableWaterContent];
    [self deficit];
    [self percentDeficit];
    [self mADDecimal];
    [self mADLine];

    // Trying to display a value in labels for FC, Depletion, and WP.
    // % is a placeholder, .2f is a two deciminal float number
    // inserting a \" shows the inches symbol

    // Some of these labels were placed on the Storyboard and are connected here, others
are created in code.
    NSString *fcSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"fcSaved"];
    float fcValue = [fcSaved floatValue];
    NSString *fCString = [[NSString alloc] initWithFormat:@"%.2f\"", fcValue];
    //Clear for update of added irrigation
    self.fCLabel.text = @"";
    self.fCLabel.text = [self.fCLabel.text stringByAppendingString:fCString];

    NSString *pwpSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"pwpSaved"];
    float pwpValue = [pwpSaved floatValue];
    NSString *pwpString = [[NSString alloc] initWithFormat:@"%.2f\"", pwpValue];
    self.wPLabel.text = @"";
    self.wPLabel.text = [self.wPLabel.text stringByAppendingString:pwpString];

    // Create label in code
    if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad) {

        UILabel *dateLabel = [[UILabel alloc] initWithFrame:CGRectMake(110, 68, 200,
19)];
        dateLabel.textAlignment = NSTextAlignmentCenter;
        [dateLabel setFont:[UIFont fontWithName:@"HelveticaNeue-Bold" size:17.0f]];
        dateLabel.backgroundColor = [UIColor clearColor];
        dateLabel.textColor = [UIColor whiteColor];
```

```
        NSString *dateSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"dateSaved"];
        [dateLabel setText:@""];
        [dateLabel setText:dateSaved];
        [self.view addSubview:dateLabel];

    } else {

        UILabel *dateLabel = [[UILabel alloc] initWithFrame:CGRectMake(110, 68, 100,
19)];
        dateLabel.textAlignment = NSTextAlignmentCenter;
        [dateLabel setFont:[UIFont fontWithName:@"HelveticaNeue-Bold" size:17.0f]];
        dateLabel.backgroundColor = [UIColor clearColor];
        dateLabel.textColor = [UIColor whiteColor];
        NSString *dateSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"dateSaved"];
        [dateLabel setText:@""];
        [dateLabel setText:dateSaved];
        [self.view addSubview:dateLabel];

    }

    // Create label and lay on top of any existing label - for updating after add irrigation
    if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad) {

        UILabel *irrigationNeededLabel = [[UILabel alloc]
initWithFrame:CGRectMake(333, 300, 67, 77)];
        irrigationNeededLabel.textAlignment = NSTextAlignmentLeft;
        irrigationNeededLabel.lineBreakMode = NSLineBreakByWordWrapping;
        irrigationNeededLabel.numberOfLines = 4;
        [irrigationNeededLabel setFont:[UIFont fontWithName:@"HelveticaNeue-Light"
size:15.0f]];
        irrigationNeededLabel.textColor = [UIColor whiteColor];
        irrigationNeededLabel.backgroundColor = [UIColor colorWithRed:.06667
green:.2941 blue:.2118 alpha:1];

        // Place string into previous label
        NSString *irrigationNeededString = [[NSString alloc] initWithFormat:@"Net
Irrigation Needed = %.2f\"", self.deficit];//grossIrrNeeded];
        [irrigationNeededLabel setText:irrigationNeededString];
        [self.view addSubview:irrigationNeededLabel];

    } else {

        UILabel *irrigationNeededLabel = [[UILabel alloc]
initWithFrame:CGRectMake(233, 175, 67, 77)];
```

```
        irrigationNeededLabel.textAlignment = NSTextAlignmentLeft;
        irrigationNeededLabel.lineBreakMode = NSLineBreakByWordWrapping;
        irrigationNeededLabel.numberOfLines = 4;
        [irrigationNeededLabel setFont:[UIFont fontWithName:@"HelveticaNeue-Light"
size:15.0f]];
        irrigationNeededLabel.textColor = [UIColor whiteColor];
        irrigationNeededLabel.backgroundColor = [UIColor colorWithRed:.06667
green:.2941 blue:.2118 alpha:1];

        // Place string into previous label
        NSString *irrigationNeededString = [[NSString alloc] initWithFormat:@"Net
Irrigation Needed = %.2f\"", self.deficit];//grossIrrNeeded];
        [irrigationNeededLabel setText:irrigationNeededString];
        [self.view addSubview:irrigationNeededLabel];
    }

    if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad) {
        UILabel *mAD1Label = [[UILabel alloc] initWithFrame:CGRectMake(22,
((self.mADDecimal) * 500 + 70), 58, 48)];
        mAD1Label.textAlignment = NSTextAlignmentRight;
        mAD1Label.lineBreakMode = NSLineBreakByWordWrapping;
        mAD1Label.numberOfLines = 2;
        [mAD1Label setFont:[UIFont fontWithName:@"HelveticaNeue-Light" size:17.0f]];
        //RGB Color is the number divided by 255
        mAD1Label.textColor = [UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1];
        mAD1Label.backgroundColor = [UIColor clearColor];

        NSString *awcSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"awcSaved"];
        NSString *mADNumberString1 = [[NSString alloc] initWithFormat:@"MAD =
%.2f\"", (((1 - self.mADDecimal) * [awcSaved floatValue]) + [pwpSaved floatValue])];
        NSLog(@"MADDecimal: %.2f, AWC: %@, PWP: %@", self.mADDecimal,
awcSaved, pwpSaved);
        [mAD1Label setText:@""];
        mAD1Label.text = nil;
        [mAD1Label setText:mADNumberString1];
        [self.view addSubview:mAD1Label];

    } else {
        UILabel *mAD1Label = [[UILabel alloc] initWithFrame:CGRectMake(22,
((self.mADDecimal) * 250 + 70), 58, 48)];
        mAD1Label.textAlignment = NSTextAlignmentRight;
        mAD1Label.lineBreakMode = NSLineBreakByWordWrapping;
        mAD1Label.numberOfLines = 2;
        [mAD1Label setFont:[UIFont fontWithName:@"HelveticaNeue-Light" size:17.0f]];
        //RGB Color is the number divided by 255
```

```
        mAD1Label.textColor = [UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1];
        mAD1Label.backgroundColor = [UIColor clearColor];

        NSString *awcSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"awcSaved"];
        NSString *mADNumberString1 = [[NSString alloc] initWithFormat:@"MAD =
%.2f\"", (((1 - self.mADDecimal) * [awcSaved floatValue]) + [pwpSaved floatValue])];
        NSLog(@"MADDecimal: %.2f, AWC: %@, PWP: %@", self.mADDecimal,
awcSaved, pwpSaved);
        [mAD1Label setText:@""];
        mAD1Label.text = nil;
        [mAD1Label setText:mADNumberString1];
        [self.view addSubview:mAD1Label];
    }

    if (self.isViewLoaded) {
        // setNeedsDisplay redraws the profile gauge - WaterGauge class
        [self.gaugeView setNeedsDisplay];
        // Animates the gauge value change
        [UIView transitionWithView:gaugeView duration:1
options:UIViewAnimationOptionTransitionCrossDissolve
animations:^{[gaugeView.layer displayIfNeeded];} completion:nil];

        [self.weatheriPadView setNeedsDisplay];
        [UIView transitionWithView:weatheriPadView duration:1
options:UIViewAnimationOptionTransitionCrossDissolve
animations:^{[weatheriPadView.layer displayIfNeeded];} completion:nil];
    }

    // GDD % bar ---- Currently placed in waterweatheripadview
    if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad) {

        self.percGrowth =[[NSUserDefaults standardUserDefaults]
stringForKey:@"percGrowthSaved"];

        // GDD bar with text inside
        UILabel *GDDBar = [[UILabel alloc]
initWithFrame:CGRectMake(40,700,688,25)];
        [GDDBar setFont:[UIFont boldSystemFontOfSize:12]];
        GDDBar.textColor = [UIColor lightTextColor];
        GDDBar.backgroundColor = [UIColor colorWithRed:.06666667 green:.55
blue:.2117647 alpha:1];
        GDDBar.textAlignment = NSTextAlignmentCenter;
        [GDDBar setText:@"0% ---Percent Growth--- 100%"];
        GDDBar.layer.borderColor = [UIColor blackColor].CGColor;
        GDDBar.layer.borderWidth = 1.0;
```

```
        GDDBar.layer.cornerRadius = 5;
        GDDBar.clipsToBounds = YES;

        // Get the percent growth and tell where to put on GDD bar
        float percentGrowthLine = [percGrowth floatValue] * 688 + 20;
        NSLog(@"per Growthline = %f, %f", percentGrowthLine, [percGrowth
floatValue]);

        // Create line
        UIView *GDDLine = [[UIView alloc]
initWithFrame:CGRectMake(percentGrowthLine, 700, 2, 35)];
        GDDLine.backgroundColor = [UIColor blackColor];

        //Center the text bubble, 15 for half-width, 1 more for line size.
        float percentGrowthValueLine = percentGrowthLine - 14;

        // Place circle below line with value of percent GDD
        UILabel *GDDPercCircle = [[UILabel alloc]
initWithFrame:CGRectMake(percentGrowthValueLine, 735, 30, 30)];
        [GDDPercCircle setFont:[UIFont boldSystemFontOfSize:10]];
        GDDPercCircle.textColor = [UIColor blackColor];
        GDDPercCircle.backgroundColor = [UIColor lightTextColor];
        GDDPercCircle.textAlignment = NSTextAlignmentCenter;
        GDDPercCircle.layer.cornerRadius = 15.0;
        GDDPercCircle.layer.masksToBounds = YES;
        GDDPercCircle.text = [NSString stringWithFormat:@"%.0f%%", ([self.percGrowth
floatValue] *100)];
    }
}

- (float)availableWaterContent
{
    NSString *fcSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"fcSaved"];
    NSString *pwpSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"pwpSaved"];

    // Convert saved string to float variables
    float fc = [fcSaved floatValue];
    float pwp = [pwpSaved floatValue];
    NSLog(@"fc: %f, pwp: %f", fc, pwp);

    float awc = fc - pwp;
    return awc;
}
```

```
- (float)deficit
{
    NSString *deficitSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"deficitCurrentSaved"];
    float deficitCurrent = [deficitSaved floatValue];
    deficitCurrent = (deficitCurrent * -1);
    NSLog(@"Deficit Current: %f", deficitCurrent);
    return deficitCurrent;
}

// Calculate percent deficit and convert to points on screen
// This will determine how large the red rectancle will be from top down - used in
WaterGauge class
- (float) percentDeficit
{
    NSLog(@"Percent Deficit: %f", ((self.deficit / (self.availableWaterContent)) *250));
    return ((self.deficit / (self.availableWaterContent)) * 250);
}

- (float)mADDecimal
{
    NSString *pawMADSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"pawAtMadSaved"];
    NSString *awcSaved = [[NSUserDefaults standardUserDefaults]
stringForKey:@"awcSaved"];
    float mADDecimal = 1 - ([pawMADSaved floatValue] / [awcSaved floatValue]);

    return mADDecimal;
}

// Convert MAD percent to a line on the gauge
- (float)mADLine
{
    // Gauge is 250 pixel high
    NSLog(@"MADLine: %f", (self.mADDecimal *250));
    return (self.mADDecimal);
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

// Done Button returns addIrrigation view to summary view, and uploads irr to eRAMS,
and updates summary view
- (IBAction)done:(UIStoryboardSegue *)segue
```

```
{
    //segue to summary view
    WaterAddIrrigationViewController *userInput = [segue sourceViewController];

    // Bring variables that user inputted in addIrr view
    NSDate *date = userInput.datePicked.date;
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    [formatter setDateFormat:@"yyyy-MM-dd"];
    NSString *dateString = [formatter stringFromDate:date];
    NSLog(@"Date selected: %@, Irrigation Entered: %@, Precip Entered: %@",
dateString, userInput.irrigationAmountEntered.text,
userInput.precipiationAmountEntered.text);

    NSString *savedToken = [[NSUserDefaults standardUserDefaults]
stringForKey:@"tokenName"];
    NSString *currentProjectIDNum = [[NSUserDefaults standardUserDefaults]
stringForKey:@"currentProjectID"];
    NSString *gid = [[NSUserDefaults standardUserDefaults]
stringForKey:@"gidSaved"];
    NSString *irrigationGrossEntered = userInput.irrigationAmountEntered.text;
    NSString *precipitationEntered = userInput.precipiationAmountEntered.text;
    NSString *soilMoistureEntered = userInput.soilMoistureEntered.text;

    if ([soilMoistureEntered  isEqual: @""]) {
       soilMoistureEntered = soilMoistureEntered;
    } else {
    NSString *decimal = @"0.";
    soilMoistureEntered = [decimal stringByAppendingString:soilMoistureEntered];
    }

    if (![irrigationGrossEntered isEqualToString:@""]) {

    NSString *URLStringIrr = [[NSString alloc]
initWithFormat:@"https://erams.com/api/%@/irrigation/enterIrrigation", savedToken];

    // create our URL based on the string
    NSURL *baseURLIrr = [NSURL URLWithString:URLStringIrr];
    NSLog(@"Post URL: %@", baseURLIrr);

    AFHTTPClient *httpClient = [[AFHTTPClient alloc] initWithBaseURL:baseURLIrr];

    NSDictionary *parameter = @{@"gid":     gid,
                                @"proj":    currentProjectIDNum,
                                @"date":    dateString,
                                @"amount":  irrigationGrossEntered
                                };
```

```
    NSLog(@"gid: %@, proj: %@, date: %@, amount: %@", gid, currentProjectIDNum,
dateString, irrigationGrossEntered);
    [httpClient setParameterEncoding:AFJSONParameterEncoding];
    [httpClient registerHTTPOperationClass:[AFJSONRequestOperation class]];

    [httpClient getPath:URLStringIrr parameters:parameter
success:^(AFHTTPRequestOperation *operation, id responseObject) {
        // Print the response body in text
        NSLog(@"Params: %@", parameter);
        NSLog(@"Response: %@",responseObject);

        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:nil message:@"Add
Irrigation Successful!" delegate:self cancelButtonTitle:nil otherButtonTitles:nil];
        [alert show];
        [self performSelector:@selector(dismissAlert:) withObject:alert afterDelay:3.0f];

        [gaugeSpinner startAnimating];
        [self viewDidLoad];

    } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
        NSLog(@"FAIL");
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"ERROR"
message:@"Add Irrigation Failed!" delegate:self cancelButtonTitle:nil
otherButtonTitles:nil];
        [alert show];
        [self performSelector:@selector(dismissAlert:) withObject:alert afterDelay:3.0f];
    }];
    }

    // If the user entered a precip value, send it to the cloud
    if (![precipitationEntered isEqualToString:@""]) {

        NSString *URLStringPrecip = [[NSString alloc]
initWithFormat:@"https://erams.com/api/%@/irrigation/enterPrecipitation",
savedToken];

        // create our URL based on the string
        NSURL *baseURLPrecip = [NSURL URLWithString:URLStringPrecip];
        NSLog(@"Post URL: %@", baseURLPrecip);

        AFHTTPClient *httpClient = [[AFHTTPClient alloc]
initWithBaseURL:baseURLPrecip];

        NSDictionary *parameter = @{@"gid":     gid,
                                    @"proj":    currentProjectIDNum,
                                    @"date":    dateString,
```

```
                                @"amount":  precipitationEntered
                                };
        NSLog(@"gid: %@, proj: %@, date: %@, amount: %@", gid,
currentProjectIDNum, dateString, precipitationEntered);
        [httpClient setParameterEncoding:AFJSONParameterEncoding];
        [httpClient registerHTTPOperationClass:[AFJSONRequestOperation class]];

        [httpClient getPath:URLStringPrecip parameters:parameter
success:^(AFHTTPRequestOperation *operation, id responseObject) {
            // Print the response body in text
            NSLog(@"Params: %@", parameter);
            NSLog(@"Response: %@",responseObject);

            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:nil message:@"Add
Precipitation Successful!" delegate:self cancelButtonTitle:nil otherButtonTitles:nil];
            //Shift the alert lower
            alert.transform = CGAffineTransformTranslate(alert.transform, 0.0, 100.0);
            [alert show];
            [self performSelector:@selector(dismissAlert:) withObject:alert afterDelay:3.0f];

            [gaugeSpinner startAnimating];
            [self viewDidLoad];

        } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
            NSLog(@"FAIL");
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"ERROR"
message:@"Add Precipitation Failed!" delegate:self cancelButtonTitle:nil
otherButtonTitles:nil];
            [alert show];
            [self performSelector:@selector(dismissAlert:) withObject:alert afterDelay:3.0f];
        }];
    }

    // If the user entered an observed soil moisture, send it to the cloud
    if (![soilMoistureEntered isEqualToString:@""]) {

        NSString *URLStringSoilMoisture = [[NSString alloc]
initWithFormat:@"https://erams.com/api/%@/irrigation/enterObservedSoilMoisture",
savedToken];

        // create our URL based on the string
        NSURL *baseURLSoilMoisture = [NSURL
URLWithString:URLStringSoilMoisture];
        NSLog(@"Post URL: %@", baseURLSoilMoisture);

        AFHTTPClient *httpClient = [[AFHTTPClient alloc]
```

```
initWithBaseURL:baseURLSoilMoisture];

        NSDictionary *parameter = @{@"gid":     gid,
                                    @"proj":    currentProjectIDNum,
                                    @"date":    dateString,
                                    @"amount":  soilMoistureEntered
                                    };
        NSLog(@"gid: %@, proj: %@, date: %@, amount: %@", gid,
currentProjectIDNum, dateString, soilMoistureEntered);
        [httpClient setParameterEncoding:AFJSONParameterEncoding];
        [httpClient registerHTTPOperationClass:[AFJSONRequestOperation class]];

        [httpClient getPath:URLStringSoilMoisture parameters:parameter
success:^(AFHTTPRequestOperation *operation, id responseObject) {
            // Print the response body in text
            NSLog(@"Params: %@", parameter);
            NSLog(@"Response: %@",responseObject);

            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:nil message:@"Add Soil
Moisture Successful!" delegate:self cancelButtonTitle:nil otherButtonTitles:nil];
            [alert show];
            [self performSelector:@selector(dismissAlert:) withObject:alert afterDelay:3.0f];

            [gaugeSpinner startAnimating];
            [self viewDidLoad];

        } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
            NSLog(@"FAIL");
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"ERROR"
message:@"Add Soil Moisture Failed!" delegate:self cancelButtonTitle:nil
otherButtonTitles:nil];
            [alert show];
            [self performSelector:@selector(dismissAlert:) withObject:alert afterDelay:3.0f];
        }];
    }
}

// Dismiss the success/failure of add irrigation without the user pressing a button
-(void)dismissAlert:(UIAlertView *) alertView
{
    [alertView dismissWithClickedButtonIndex:nil animated:YES];
}

// Dismiss info view
- (IBAction)returnInformation:(UIStoryboardSegue *)segue
{
```

```
    if ([[segue identifier] isEqualToString:@"returnInformation"]) {
        [self dismissViewControllerAnimated:YES completion:NULL];
    }
}

// Cancel add irrigation view
- (IBAction)cancelAddIrrigation:(UIStoryboardSegue *)segue
{
    [segue sourceViewController];
}

// Add Irr button
- (IBAction)addButton:(id)sender {
}

// Check weather button
- (IBAction)weatherButton:(id)sender {
}
@end
```

```
//  WaterGauge.h

#import <UIKit/UIKit.h>

@interface WaterGauge : UIView

@end
```

```
//  WaterGauge.m

#import "WaterGauge.h"
#import "WaterViewController.h"

@interface WaterGauge()

// This connects this class view controller data
@property (nonatomic, strong) WaterViewController *data;

@end

@implementation WaterGauge

// Sythesize data from WaterViewController
@synthesize data = _data;
```

```
- (WaterViewController *)data
{
    if (!_data) _data = [[WaterViewController alloc] init];
    return _data;
}

- (void)setup
{
    self.contentMode = UIViewContentModeRedraw;
}

-(void)awakeFromNib
{
    [self setup];
}

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        [self setup];
    }
    return self;
}

// Calling drawRect
- (void)drawRect:(CGRect)rect
{

    // For iPad
    if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad) {

        CGContextRef Context = UIGraphicsGetCurrentContext();

        // This is the full water content
        // Blue box
        CGContextSetRGBFillColor(Context, 0, .27, .53, 1.0);
        CGContextFillRect(Context, CGRectMake(1, 1, 200, 500));

        CGContextRef RedBar = UIGraphicsGetCurrentContext();

        // This is the red deficit bar
        CGContextSetRGBFillColor(RedBar, .5, 0, 0, 1.0);
        CGContextFillRect(RedBar, CGRectMake(1, 1, 200, /*the height is variable*/
(self.data.percentDeficit * 2)));
        CGContextSetLineWidth(RedBar, 3.0);
```

```
        [[UIColor blueColor] setStroke];

        // Draw lines around the gauge
        CGContextSetRGBStrokeColor(Context, 0, 0, 0, 1);
        CGContextSetLineWidth(Context, 4);
        CGPoint corners[8] = { CGPointMake(0, 0), CGPointMake(200, 0),
CGPointMake(200, 0), CGPointMake(200, 500), CGPointMake(200, 500),
CGPointMake(0, 500), CGPointMake(0, 500), CGPointMake(0, 0)};
        CGContextStrokeLineSegments(Context, corners, 8);

        CGContextRef MadLine = UIGraphicsGetCurrentContext();
        // Draw a black line for MAD
        CGContextSetRGBStrokeColor(MadLine, 0, 0, 0, .8); // Color black, 20%
transparent
        CGContextSetLineWidth(MadLine, 4);
        // x points static, y is variable
        // *2 since iPad bar is twice as tall
        CGPoint points[2] = { CGPointMake(0, (self.data.mADLine * 500)),
CGPointMake(200, (self.data.mADLine * 500))};
        CGContextStrokeLineSegments(MadLine, points, 2);

    } else {

    CGContextRef Context = UIGraphicsGetCurrentContext();

    // This is the full water content
    // Blue box
    CGContextSetRGBFillColor(Context, 0, .27, .53, 1.0);
    CGContextFillRect(Context, CGRectMake(1, 1, 100, 250));

    CGContextRef RedBar = UIGraphicsGetCurrentContext();

    // This is the red deficit bar
    CGContextSetRGBFillColor(RedBar, .5, 0, 0, 1.0);
    CGContextFillRect(RedBar, CGRectMake(1, 1, 100, /*the height is variable*/
self.data.percentDeficit));
    CGContextSetLineWidth(RedBar, 3.0);
    [[UIColor blueColor] setStroke];

    // Draw lines around the gauge
    CGContextSetRGBStrokeColor(Context, 0, 0, 0, 1);
    CGContextSetLineWidth(Context, 4);
    CGPoint corners[8] = { CGPointMake(0, 0), CGPointMake(100, 0),
CGPointMake(100, 0), CGPointMake(100, 250), CGPointMake(100, 250),
CGPointMake(0, 250), CGPointMake(0, 250), CGPointMake(0, 0)};
    CGContextStrokeLineSegments(Context, corners, 8);
```

```
    CGContextRef MadLine = UIGraphicsGetCurrentContext();
    // Draw a black line for MAD
    CGContextSetRGBStrokeColor(MadLine, 0, 0, 0, .8); // Color black, 20% transparent
    CGContextSetLineWidth(MadLine, 4);
    // x points static, y is variable
    CGPoint points[2] = { CGPointMake(0, (self.data.mADLine *250)),
CGPointMake(100, (self.data.mADLine * 250))};
    CGContextStrokeLineSegments(MadLine, points, 2);
    }
}

@end
```

```
//  RKAttributes.h

#import <Foundation/Foundation.h>

@interface RKAttributes : NSObject

@property (nonatomic, copy) NSString *awc;
@property (nonatomic, copy) NSString *fc;
@property (nonatomic, copy) NSString *pwp;
@property (nonatomic, copy) NSString *paw;
@property (nonatomic, copy) NSString *pawAtMad;
@property (nonatomic, copy) NSString *dateValue;
@property (nonatomic, copy) NSString *etc;
@property (nonatomic, copy) NSString *gdd;
@property (nonatomic, copy) NSString *deficitCurrent;
@property (nonatomic, copy) NSString *etr;
@property (nonatomic, copy) NSString *percentGrowth;
@property (nonatomic, copy) NSString *depthRootZone;
@property (nonatomic, copy) NSString *precipitation;
@property (nonatomic, copy) NSString *irrigationFromWeb;
@property (nonatomic, copy) NSString *kcr;
@property (nonatomic, copy) NSString *tempMax;
@property (nonatomic, copy) NSString *ks;
@property (nonatomic, copy) NSString *defMad;
@property (nonatomic, copy) NSString *tempMin;

@end
```

```
//  RKAttributes.m
```

```
#import "RKAttributes.h"

@implementation RKAttributes

- (NSString *)description
{
    return [NSString stringWithFormat:@"%@, %@ ",self.dateValue, self.gdd];
}

@end
```

```
//  WaterAddIrrigationViewController.h

#import <UIKit/UIKit.h>

@interface WaterAddIrrigationViewController : UIViewController
<UITextFieldDelegate>

// Values user would like to upload to eRAMS
@property (strong, nonatomic) IBOutlet UITextField *irrigationAmountEntered;
@property (strong, nonatomic) IBOutlet UITextField *precipiationAmountEntered;
@property (strong, nonatomic) IBOutlet UITextField *soilMoistureEntered;
@property (strong, nonatomic) IBOutlet UIDatePicker *datePicked;
@property (strong, nonatomic) WaterAddIrrigationViewController *irrigationAdded;

@end
```

```
//  WaterAddIrrigationViewController.m

#import "WaterAddIrrigationViewController.h"
#import "WaterCalculateViewController.h"

@interface WaterAddIrrigationViewController ()
@end

@implementation WaterAddIrrigationViewController

@synthesize irrigationAmountEntered, precipiationAmountEntered,
soilMoistureEntered, datePicked;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
```

```
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    // Date picker background color, light green
    datePicked.backgroundColor = [UIColor lightTextColor];
    self.navigationController.navigationBar.barStyle = UIBarStyleBlackOpaque;

    // Show alert on initial view of add irrigation
    if (![[NSUserDefaults standardUserDefaults] boolForKey:@"HasAddedIrrOnce"]) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle: @"Warning"
                                                        message:@"Only upload 1 value at a time to ensure
proper upload."
                                                       delegate:self
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
        [alert show];
        [[NSUserDefaults standardUserDefaults] setBool:YES
forKey:@"HasAddedIrrOnce"];
    } else {
        //Seen addIrr view already, no need to show alert
    }
}

- (void)doneWithNumberPad
{
    // Dismiss keyboard
    [irrigationAmountEntered resignFirstResponder];
    [precipiationAmountEntered resignFirstResponder];
    [soilMoistureEntered resignFirstResponder];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

// Create labels on keyboard
-(void)textFieldDidBeginEditing:(UITextField *)textField
{
    textField.keyboardAppearance = UIKeyboardAppearanceDark;
```

```
    UIToolbar* numberToolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel = [[UIBarButtonItem alloc]
initWithTitle:@"Irrigation in inches" style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];
    numberToolbar.items = [NSArray arrayWithObjects:textfieldLabel, flexibleSpace,
doneButton, nil];
    [numberToolbar sizeToFit];
    [textfieldLabel setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == irrigationAmountEntered) {
        irrigationAmountEntered.inputAccessoryView = numberToolbar;
    }

    UIToolbar* numberToolbar2 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar2.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel2 = [[UIBarButtonItem alloc]
initWithTitle:@"Precipitation in inches" style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace2 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton2 = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];
    numberToolbar2.items = [NSArray arrayWithObjects:textfieldLabel2, flexibleSpace2,
doneButton2, nil];
    [numberToolbar2 sizeToFit];
    [textfieldLabel2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == precipiationAmountEntered) {
        precipiationAmountEntered.inputAccessoryView = numberToolbar2;
    }

    UIToolbar* numberToolbar3 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar3.barStyle = UIBarStyleBlack;
```

```
    UIBarButtonItem *textfieldLabel3 = [[UIBarButtonItem alloc]
initWithTitle:@"Observed Soil Deficit %" style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace3 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton3 = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];
    numberToolbar3.items = [NSArray arrayWithObjects:textfieldLabel3, flexibleSpace3,
doneButton3, nil];
    [numberToolbar3 sizeToFit];
    [textfieldLabel3 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton3 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == soilMoistureEntered) {
        //show alert

        // Let user know that if they enter observed soil deficit, it will override current value
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Observed Soil Deficit"
                                                        message:@"Entering observed soil deficit information
will override your current calculated soil deficit."
                                                       delegate:self
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
        [alert show];
        soilMoistureEntered.inputAccessoryView = numberToolbar3;
    }
}

// Activates the return button on the keyboard
- (BOOL)textFieldShouldReturn:(UITextField *)theTextField {
    if ((theTextField == self.irrigationAmountEntered) || (theTextField ==
self.precipiationAmountEntered)) {
        [theTextField resignFirstResponder];
    }
    return YES;
}

// Done button from calculator
// Takes value from label in calculator and inserts it into irrigation amount
- (IBAction)doneCalculating:(UIStoryboardSegue *)segue
{
    WaterCalculateViewController *irrigationCalculated = [segue sourceViewController];

    //Remove " (inches) from calcuation
    NSString *calculation = irrigationCalculated.answerLabel.text;
```

```
    NSString *calculationNumber = [calculation substringToIndex:[calculation length] -1];

    //Set text box equal to amount calculated in previous view
    self.irrigationAmountEntered.text = [NSString stringWithFormat:@"%@",
calculationNumber];
}

@end
```

```
//  RKAddIrrigation.h

#import <Foundation/Foundation.h>

@interface RKAddIrrigation : NSObject

@property (nonatomic, copy) NSString *gidToSend;
@property (nonatomic, copy) NSString *projectCurrentToSend;
@property (nonatomic, copy) NSString *dateToSend;
@property (nonatomic, copy) NSString *irrigationGrossToSend;

@property (nonatomic, copy) NSString *message;
@property (nonatomic, copy) NSString *name;
@property (nonatomic, copy) NSString *details;
@end
```

```
//  RKAddIrrigation.m

#import "RKAddIrrigation.h"
@implementation RKAddIrrigation
@end
```

```
//  RKAddIrrResponse.h

#import <Foundation/Foundation.h>

@interface RKAddIrrResponse : NSObject

@property (nonatomic, copy) NSArray *message;
@property (nonatomic, copy) NSString *name;
@property (nonatomic, copy) NSString *details;

@end
```

```
//  RKAddIrrResponse.m

#import "RKAddIrrResponse.h"

@implementation RKAddIrrResponse

@synthesize message, name, details;

- (NSString *)description
{
   NSLog(@"Response is: %@, %@, %@", self.message, self.name, self.details);
   return self.name;
}
@end
```

```
//  WaterCalculateViewController.h

#import <UIKit/UIKit.h>

@interface WaterCalculateViewController : UIViewController <UITextFieldDelegate>

// Text fields to enter calculation parameters
@property (weak, nonatomic) IBOutlet UITextField *cFS;
@property (weak, nonatomic) IBOutlet UITextField *gPM;
@property (weak, nonatomic) IBOutlet UITextField *hoursIrrigated;
@property (weak, nonatomic) IBOutlet UITextField *acresIrrigated;
@property (weak, nonatomic) IBOutlet UILabel *answerLabel;
@property (weak, nonatomic) IBOutlet UIButton *calculateButton;

// Output of float variables
@property (nonatomic) float cFSToInches;
@property (nonatomic) float gPMToInches;

// Calculate button
-(IBAction)calculate:(id)sender;

@end
```

```
//  WaterCalculateViewController.m

#import "WaterCalculateViewController.h"
#import "WaterAddIrrigationViewController.h"
```

```
@interface WaterCalculateViewController ()

@end

@implementation WaterCalculateViewController
@synthesize cFS, gPM, hoursIrrigated, acresIrrigated, cFSToInches, gPMToInches,
answerLabel, calculateButton;

// Calculate values
-(IBAction)calculate:(id)sender {

    // Show alert if there are values in both the CFS and GPM
    if (((![cFS.text length]) == 0) && ((![gPM.text length]) == 0)) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle: @"Values"
                                                        message:@"Please don't enter a value for both CFS
and GPM."
                                                       delegate:self
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
        [alert show];

        // Calculates the CFS to inches
        // If no value in hours or acres, show alert
    } else if (([hoursIrrigated.text length] == 0) || ([acresIrrigated.text length] == 0)) {

        UIAlertView *alert2 = [[UIAlertView alloc] initWithTitle: @"Values"
                                                         message:@"Please enter a value for either CFS or
GPM.  Also, please enter values for both Hours and Acres."
                                                        delegate:self
                                               cancelButtonTitle:@"OK"
                                               otherButtonTitles:nil];

        [alert2 show];

        // Calculate CFS to Inches
    } else if (![cFS.text length] == 0) {
        float cFSNumber = [cFS.text floatValue];
        float hoursIrrigatedNumber = [hoursIrrigated.text floatValue];
        float acresIrrigatedNumber = [acresIrrigated.text floatValue];
        cFSToInches = ((cFSNumber * (hoursIrrigatedNumber * 3600) * 12) /
(acresIrrigatedNumber * 43560));
        // Clear the label before submitting another answer
        self.answerLabel.text = @"";

        NSString *answerString = [[NSString alloc]
initWithFormat:@"%.2f\"",cFSToInches];
```

```
        self.answerLabel.text = [self.answerLabel.text
stringByAppendingString:answerString];

        // Calculates the GPM to inches
    } else if (![gPM.text length] == 0) {
        float gPMNumber = [gPM.text floatValue];
        float hoursIrrigatedNumber = [hoursIrrigated.text floatValue];
        float acresIrrigatedNumber = [acresIrrigated.text floatValue];
        gPMToInches = (((gPMNumber *0.133681) * (hoursIrrigatedNumber * 60) * 12) /
(acresIrrigatedNumber *43560));
        // Clear the label before submitting another answer
        self.answerLabel.text = @"";

        NSString *answerString = [[NSString alloc]
initWithFormat:@"%.2f\"",gPMToInches];
        self.answerLabel.text = [self.answerLabel.text
stringByAppendingString:answerString];
    }
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.navigationController.navigationBar.barStyle = UIBarStyleBlackOpaque;
}

-(void)doneWithFlow
{
    [self.hoursIrrigated becomeFirstResponder];
}

-(void)doneWithHours
{
    [self.acresIrrigated becomeFirstResponder];
}

- (void)doneWithNumberPad
{
```

```
    [self.acresIrrigated resignFirstResponder];
}

// Add labels on keyboard
-(void)textFieldDidBeginEditing:(UITextField *)textField
{
    textField.keyboardAppearance = UIKeyboardAppearanceDark;

    UIToolbar* numberToolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel = [[UIBarButtonItem alloc] initWithTitle:@"Enter
CFS" style:UIBarButtonItemStylePlain target:nil action:nil];
    UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton = [[UIBarButtonItem alloc] initWithTitle:@"Next"
style:UIBarButtonItemStyleDone target:self action:@selector(doneWithFlow)];

    numberToolbar.items = [NSArray arrayWithObjects:textfieldLabel, flexibleSpace,
doneButton, nil];
    [numberToolbar sizeToFit];
    [textfieldLabel setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == cFS) {
        cFS.inputAccessoryView = numberToolbar;
    }

    UIToolbar* numberToolbar2 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar2.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel2 = [[UIBarButtonItem alloc] initWithTitle:@"Enter
GPM" style:UIBarButtonItemStylePlain target:nil action:nil];
    UIBarButtonItem *flexibleSpace2 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton2 = [[UIBarButtonItem alloc] initWithTitle:@"Next"
style:UIBarButtonItemStyleDone target:self action:@selector(doneWithFlow)];

    numberToolbar2.items = [NSArray arrayWithObjects:textfieldLabel2, flexibleSpace2,
doneButton2, nil];
    [numberToolbar2 sizeToFit];
    [textfieldLabel2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == gPM) {
        gPM.inputAccessoryView = numberToolbar2;
```

```
    }

    UIToolbar* numberToolbar3 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar3.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel3 = [[UIBarButtonItem alloc]
initWithTitle:hoursIrrigated.placeholder style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace3 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton3 = [[UIBarButtonItem alloc] initWithTitle:@"Next"
style:UIBarButtonItemStyleDone target:self action:@selector(doneWithHours)];

    numberToolbar3.items = [NSArray arrayWithObjects:textfieldLabel3, flexibleSpace3,
doneButton3, nil];
    [numberToolbar3 sizeToFit];
    [textfieldLabel3 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton3 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == hoursIrrigated) {
        hoursIrrigated.inputAccessoryView = numberToolbar3;
    }

    UIToolbar* numberToolbar4 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar4.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel4 = [[UIBarButtonItem alloc]
initWithTitle:acresIrrigated.placeholder style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace4 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton4 = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];

    numberToolbar4.items = [NSArray arrayWithObjects:textfieldLabel4, flexibleSpace4,
doneButton4, nil];
    [numberToolbar4 sizeToFit];
    [textfieldLabel4 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton4 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == acresIrrigated) {
        acresIrrigated.inputAccessoryView = numberToolbar4;
    }
}
```

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}
@end
```

```
//  WaterCalculateiPadView.h

#import <UIKit/UIKit.h>

@interface WaterCalculateiPadView : UIView

// Text fields to enter calculation parameters
@property (weak, nonatomic) IBOutlet UITextField *cFS;
@property (weak, nonatomic) IBOutlet UITextField *gPM;
@property (weak, nonatomic) IBOutlet UITextField *hoursIrrigated;
@property (weak, nonatomic) IBOutlet UITextField *acresIrrigated;
@property (weak, nonatomic) IBOutlet UILabel *answerLabel;
@property (weak, nonatomic) IBOutlet UIButton *calculateButton;

// Output of float variables
@property (nonatomic) float cFSToInches;
@property (nonatomic) float gPMToInches;

// Calculate button
-(IBAction)calculate:(id)sender;

@end
```

```
//  WaterCalculateiPadView.m

#import "WaterCalculateiPadView.h"

@implementation WaterCalculateiPadView

@synthesize cFS, gPM, hoursIrrigated, acresIrrigated, cFSToInches, gPMToInches,
answerLabel, calculateButton;

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code
    }
```

```
    return self;
}

-(IBAction)calculate:(id)sender {

    // Show alert if there are values in both the CFS and GPM
    if ((([cFS.text length]) == 0) && (([gPM.text length]) == 0)) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle: @"Values"
                                                        message:@"Please don't enter a value for both CFS
and GPM."
                                                       delegate:self
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
        [alert show];

        // Calculates the CFS to inches
        // If no value in hours or acres, show alert
    } else if (([hoursIrrigated.text length] == 0) || ([acresIrrigated.text length] == 0)) {

        UIAlertView *alert2 = [[UIAlertView alloc] initWithTitle: @"Values"
                                                         message:@"Please enter a value for either CFS or
GPM.  Also, please enter values for both Hours and Acres."
                                                        delegate:self
                                               cancelButtonTitle:@"OK"
                                               otherButtonTitles:nil];

        [alert2 show];

        // Calculate CFS to Inches
    } else if (![cFS.text length] == 0) {
        float cFSNumber = [cFS.text floatValue];
        float hoursIrrigatedNumber = [hoursIrrigated.text floatValue];
        float acresIrrigatedNumber = [acresIrrigated.text floatValue];
        cFSToInches = ((cFSNumber * (hoursIrrigatedNumber * 3600) * 12) /
(acresIrrigatedNumber * 43560));
        // Clear the label before submitting another answer
        self.answerLabel.text = @"";

        NSString *answerString = [[NSString alloc]
initWithFormat:@"%.2f\"",cFSToInches];
        self.answerLabel.text = [self.answerLabel.text
stringByAppendingString:answerString];

        // Calculates the GPM to inches
    } else if (![gPM.text length] == 0) {
        float gPMNumber = [gPM.text floatValue];
```

```
        float hoursIrrigatedNumber = [hoursIrrigated.text floatValue];
        float acresIrrigatedNumber = [acresIrrigated.text floatValue];
        gPMToInches = (((gPMNumber *0.133681) * (hoursIrrigatedNumber * 60) * 12) /
(acresIrrigatedNumber *43560));
        // Clear the label before submitting another answer
        self.answerLabel.text = @"";

        NSString *answerString = [[NSString alloc]
initWithFormat:@"%.2f\"",gPMToInches];
        self.answerLabel.text = [self.answerLabel.text
stringByAppendingString:answerString];
    }
}

-(void)doneWithFlow
{
    [self.hoursIrrigated becomeFirstResponder];
}

-(void)doneWithHours
{
    [self.acresIrrigated becomeFirstResponder];
}

- (void)doneWithNumberPad
{
    [self.acresIrrigated resignFirstResponder];
}

// Add labels on keyboard
-(void)textFieldDidBeginEditing:(UITextField *)textField
{
    textField.keyboardAppearance = UIKeyboardAppearanceDark;

    UIToolbar* numberToolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel = [[UIBarButtonItem alloc] initWithTitle:@"Enter
CFS" style:UIBarButtonItemStylePlain target:nil action:nil];
    UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton = [[UIBarButtonItem alloc] initWithTitle:@"Next"
style:UIBarButtonItemStyleDone target:self action:@selector(doneWithFlow)];

    numberToolbar.items = [NSArray arrayWithObjects:textfieldLabel, flexibleSpace,
```

```
doneButton, nil];
    [numberToolbar sizeToFit];
    [textfieldLabel setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == cFS) {
        cFS.inputAccessoryView = numberToolbar;
    }

    UIToolbar* numberToolbar2 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar2.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel2 = [[UIBarButtonItem alloc] initWithTitle:@"Enter
GPM" style:UIBarButtonItemStylePlain target:nil action:nil];
    UIBarButtonItem *flexibleSpace2 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton2 = [[UIBarButtonItem alloc] initWithTitle:@"Next"
style:UIBarButtonItemStyleDone target:self action:@selector(doneWithFlow)];

    numberToolbar2.items = [NSArray arrayWithObjects:textfieldLabel2, flexibleSpace2,
doneButton2, nil];
    [numberToolbar2 sizeToFit];
    [textfieldLabel2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton2 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == gPM) {
        gPM.inputAccessoryView = numberToolbar2;
    }

    UIToolbar* numberToolbar3 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar3.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel3 = [[UIBarButtonItem alloc]
initWithTitle:hoursIrrigated.placeholder style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace3 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton3 = [[UIBarButtonItem alloc] initWithTitle:@"Next"
style:UIBarButtonItemStyleDone target:self action:@selector(doneWithHours)];

    numberToolbar3.items = [NSArray arrayWithObjects:textfieldLabel3, flexibleSpace3,
doneButton3, nil];
    [numberToolbar3 sizeToFit];
    [textfieldLabel3 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton3 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == hoursIrrigated) {
```

```
        hoursIrrigated.inputAccessoryView = numberToolbar3;
    }

    UIToolbar* numberToolbar4 = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
320, 50)];
    numberToolbar4.barStyle = UIBarStyleBlack;
    UIBarButtonItem *textfieldLabel4 = [[UIBarButtonItem alloc]
initWithTitle:acresIrrigated.placeholder style:UIBarButtonItemStylePlain target:nil
action:nil];
    UIBarButtonItem *flexibleSpace4 = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil
action:nil];
    UIBarButtonItem *doneButton4 = [[UIBarButtonItem alloc] initWithTitle:@"Hide
Keyboard" style:UIBarButtonItemStyleDone target:self
action:@selector(doneWithNumberPad)];

    numberToolbar4.items = [NSArray arrayWithObjects:textfieldLabel4, flexibleSpace4,
doneButton4, nil];
    [numberToolbar4 sizeToFit];
    [textfieldLabel4 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    [doneButton4 setTintColor:[UIColor colorWithRed:.72 green:.55 blue:.15 alpha:1]];
    if (textField == acresIrrigated) {
        acresIrrigated.inputAccessoryView = numberToolbar4;
    }
}

- (BOOL)textFieldShouldReturn:(UITextField *)theTextField {

    // Enables the 'next' button on the keyboard
    if (theTextField == cFS) {
        [hoursIrrigated becomeFirstResponder];
    }

    if (theTextField == gPM) {
        [hoursIrrigated becomeFirstResponder];
    }
    if (theTextField == hoursIrrigated) {
        [acresIrrigated becomeFirstResponder];
    }

    // Enables the 'Go' button and dismisses the keyboard
    if (theTextField == acresIrrigated) {
        [acresIrrigated resignFirstResponder];
        [calculateButton sendActionsForControlEvents:UIControlEventTouchUpInside];
    }
    return YES;
```

```
}

@end
```

```
//  WaterInformationViewController.h

#import <UIKit/UIKit.h>

@interface WaterInformationViewController : UIViewController

// ScrollView on the bottom half of the view
@property (weak, nonatomic) IBOutlet UIScrollView *infoScrollView;

@end
```

```
//  WaterInformationViewController.m

#import "WaterInformationViewController.h"

@interface WaterInformationViewController ()

@end

@implementation WaterInformationViewController
@synthesize infoScrollView;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    self.navigationController.navigationBar.barStyle = UIBarStyleBlackOpaque;

    // Scrollview on bottom-half of screen
    CGSize scrollViewContentSize = CGSizeMake(280, 360);
    [self.infoScrollView setContentSize:scrollViewContentSize];
```

```
    [self.infoScrollView flashScrollIndicators];

    // If additional lines are added, 20 points of height need to be added as well. This will
effect the remaining text positions for the rest of the labels.
    // Only the definitions will scroll on the screen, the 'gauge' will remain on the top
portion of the screen.

    // Tells user to scroll for more info
    UILabel *scrollLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 5, 280, 20)];
    [scrollLabel setFont:[UIFont systemFontOfSize:16]];
    scrollLabel.textAlignment = NSTextAlignmentCenter;
    scrollLabel.textColor = [UIColor blackColor];
    scrollLabel.backgroundColor = [UIColor clearColor];
    scrollLabel.lineBreakMode = NSLineBreakByWordWrapping;
    scrollLabel.numberOfLines = 0;
    [scrollLabel setText:@"Scroll for more information"];

    // Field Capacity -
    UILabel *fCLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 40, 130, 20)];
    [fCLabel setFont:[UIFont boldSystemFontOfSize:16]];
    fCLabel.textColor = [UIColor blackColor];
    fCLabel.backgroundColor = [UIColor clearColor];
    fCLabel.lineBreakMode = NSLineBreakByWordWrapping;
    fCLabel.numberOfLines = 0;
    [fCLabel setText:@"Field Capacity - "];

    // FC definition
    UILabel *fCLabelDef = [[UILabel alloc] initWithFrame:CGRectMake(150, 40, 150,
60)];
    [fCLabelDef setFont:[UIFont systemFontOfSize:16]];
    fCLabelDef.textColor = [UIColor blackColor];
    fCLabelDef.backgroundColor = [UIColor clearColor];
    fCLabelDef.lineBreakMode = NSLineBreakByWordWrapping;
    fCLabelDef.numberOfLines = 0;
    [fCLabelDef setText:@"The maximum amount of water held by soil pores."];

    // Managed Allowed Depletion -
    UILabel *madLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 110, 130,
60)];
    [madLabel setFont:[UIFont boldSystemFontOfSize:16]];
    madLabel.textColor = [UIColor blackColor];
    madLabel.backgroundColor = [UIColor clearColor];
    madLabel.lineBreakMode = NSLineBreakByWordWrapping;
    madLabel.numberOfLines = 0;
    [madLabel setText:@"Managed Allowed Depletion - "];
```

```
    // MAD definition
    UILabel *madLabelDef = [[UILabel alloc] initWithFrame:CGRectMake(150, 110,
150, 100)];
    [madLabelDef setFont:[UIFont systemFontOfSize:16]];
    madLabelDef.textColor = [UIColor blackColor];
    madLabelDef.backgroundColor = [UIColor clearColor];
    madLabelDef.lineBreakMode = NSLineBreakByWordWrapping;
    madLabelDef.numberOfLines = 0;
    [madLabelDef setText:@"Desired soil moisture deficit before stress occurs; can vary
with growth stage."];

    // Irrigation Needed Days -
    UILabel *irrLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 220, 130, 40)];
    [irrLabel setFont:[UIFont boldSystemFontOfSize:16]];
    irrLabel.textColor = [UIColor blackColor];
    irrLabel.backgroundColor = [UIColor clearColor];
    irrLabel.lineBreakMode = NSLineBreakByWordWrapping;
    irrLabel.numberOfLines = 0;
    [irrLabel setText:@"Irrigation  Needed - "];

    // Gross Irrigation needed
    UILabel *irrLabelDef = [[UILabel alloc] initWithFrame:CGRectMake(150, 220, 150,
60)];
    [irrLabelDef setFont:[UIFont systemFontOfSize:16]];
    irrLabelDef.textColor = [UIColor blackColor];
    irrLabelDef.backgroundColor = [UIColor clearColor];
    irrLabelDef.lineBreakMode = NSLineBreakByWordWrapping;
    irrLabelDef.numberOfLines = 0;
    [irrLabelDef setText:@"Net irrigation required to reach Field Capacity."];

    // Wilting Point -
    UILabel *wpLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 290, 130, 20)];
    [wpLabel setFont:[UIFont boldSystemFontOfSize:16]];
    wpLabel.textColor = [UIColor blackColor];
    wpLabel.backgroundColor = [UIColor clearColor];
    wpLabel.lineBreakMode = NSLineBreakByWordWrapping;
    wpLabel.numberOfLines = 0;
    [wpLabel setText:@"Wilting Point - "];

    // WP definition
    UILabel *wpLabelDef = [[UILabel alloc] initWithFrame:CGRectMake(150, 290, 150,
60)];
    [wpLabelDef setFont:[UIFont systemFontOfSize:16]];
    wpLabelDef.textColor = [UIColor blackColor];
    wpLabelDef.backgroundColor = [UIColor clearColor];
    wpLabelDef.lineBreakMode = NSLineBreakByWordWrapping;
```

```
    wpLabelDef.numberOfLines = 0;
    [wpLabelDef setText:@"Soil moisture where plants will permanently wilt."];

    // Add all labels to scrollview
    [self.infoScrollView addSubview:scrollLabel];
    [self.infoScrollView addSubview:fCLabel];
    [self.infoScrollView addSubview:fCLabelDef];
    [self.infoScrollView addSubview:madLabel];
    [self.infoScrollView addSubview:madLabelDef];
    [self.infoScrollView addSubview:irrLabel];
    [self.infoScrollView addSubview:irrLabelDef];
    [self.infoScrollView addSubview:wpLabel];
    [self.infoScrollView addSubview:wpLabelDef];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}
@end
```

```
//  WaterWeatherViewController.h

#import <UIKit/UIKit.h>

@interface WaterWeatherViewController : UIViewController

// ScrollView no longer used in this weather view - commented out
//@property (nonatomic, strong) IBOutlet UIScrollView *scrollView;
//@property (nonatomic, retain) IBOutlet UIPageControl *pageControl;
- (IBAction)doneWeather:(id)sender;

@end
```

```
//  WaterWeatherViewController.m

#import "WaterWeatherViewController.h"

@interface WaterWeatherViewController ()
@property (nonatomic, strong) NSString *fieldName;
@property (nonatomic, strong) NSString *projectName;
@property (nonatomic, strong) NSString *dateWeather;
@property (nonatomic, strong) NSString *etc;
@property (nonatomic, strong) NSString *gdd;
```

```
@property (nonatomic, strong) NSString *etr;
@property (nonatomic, strong) NSString *precip;
@property (nonatomic, strong) NSString *tempMax;
@property (nonatomic, strong) NSString *tempMin;
@property (nonatomic, strong) NSString *percGrowth;
@property (nonatomic, strong) NSString *ks;

@end

@implementation WaterWeatherViewController

@synthesize fieldName, projectName, dateWeather, etc, gdd, precip, tempMin,
tempMax, percGrowth, ks;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.navigationController.navigationBar.barStyle = UIBarStyleBlackOpaque;

    // Set scrollview to 320 wide * number of pages, by 381 height
    //CGSize scrollViewContentSize = CGSizeMake((320 * 1 /*pageCount number of
pages*/), 381);
    //[self.scrollView setContentSize:scrollViewContentSize];

    //Load saved variables
    self.fieldName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"fieldNameSaved"];
    self.projectName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"projectNameSaved"];
    self.dateWeather = [[NSUserDefaults standardUserDefaults]
stringForKey:@"dateWeatherSaved"];
    self.etc = [[NSUserDefaults standardUserDefaults] stringForKey:@"etcSaved"];
    self.gdd = [[NSUserDefaults standardUserDefaults] stringForKey:@"gddSaved"];
    self.etr = [[NSUserDefaults standardUserDefaults] stringForKey:@"etrSaved"];
    self.precip = [[NSUserDefaults standardUserDefaults]
stringForKey:@"precipitationSaved"];
    self.tempMax = [[NSUserDefaults standardUserDefaults]
```

```
stringForKey:@"tempMaxSaved"];
    self.tempMin =[[NSUserDefaults standardUserDefaults]
stringForKey:@"tempMinSaved"];
    self.percGrowth =[[NSUserDefaults standardUserDefaults]
stringForKey:@"percGrowthSaved"];
    self.ks =[[NSUserDefaults standardUserDefaults] stringForKey:@"ksSaved"];

    // These labels are constant
    // Previous Day ET
    UILabel *previousDayEtLabel = [[UILabel alloc] initWithFrame:CGRectMake(160,
170, 128, 21)];
    [previousDayEtLabel setFont:[UIFont systemFontOfSize:17]];
    previousDayEtLabel.textColor = [UIColor whiteColor];
    previousDayEtLabel.backgroundColor = [UIColor clearColor];
    previousDayEtLabel.textAlignment = NSTextAlignmentRight;
    [previousDayEtLabel setText:@"Daily ET"];

    // Background box
    UILabel *backgroundLabel = [[UILabel alloc] initWithFrame:CGRectMake(78, 250,
164, 191)];
    backgroundLabel.backgroundColor = [UIColor colorWithRed:0 green:0 blue:0
alpha:.4];

    UILabel *todayLabel = [[UILabel alloc] initWithFrame:CGRectMake(78, 255, 164,
21)];
    [todayLabel setFont:[UIFont boldSystemFontOfSize:17]];
    todayLabel.textColor = [UIColor whiteColor];
    todayLabel.backgroundColor = [UIColor clearColor];
    todayLabel.textAlignment = NSTextAlignmentCenter;
    [todayLabel setText:self.dateWeather];

    // Precip
    UILabel *precipLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 280, 145,
21)];
    [precipLabel setFont:[UIFont boldSystemFontOfSize:15]];
    precipLabel.textColor = [UIColor whiteColor];
    precipLabel.backgroundColor = [UIColor clearColor];
    [precipLabel setText:@"Precip"];

    // Max Temp
    UILabel *maxTempLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 302,
145, 21)];
    [maxTempLabel setFont:[UIFont boldSystemFontOfSize:15]];
    maxTempLabel.textColor = [UIColor whiteColor];
    maxTempLabel.backgroundColor = [UIColor clearColor];
    [maxTempLabel setText:@"High"];
```

```
    // Min Temp
    UILabel *minTempLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 324,
145, 21)];
    [minTempLabel setFont:[UIFont boldSystemFontOfSize:15]];
    minTempLabel.textColor = [UIColor whiteColor];
    minTempLabel.backgroundColor = [UIColor clearColor];
    [minTempLabel setText:@"Low"];

    // Total GDD
    UILabel *totalGDDLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 346,
145, 21)];
    [totalGDDLabel setFont:[UIFont boldSystemFontOfSize:15]];
    totalGDDLabel.textColor = [UIColor whiteColor];
    totalGDDLabel.backgroundColor = [UIColor clearColor];
    [totalGDDLabel setText:@"Total GDD"];

    // Perc. Growth
    UILabel *percGrowthLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 368,
145, 21)];
    [percGrowthLabel setFont:[UIFont boldSystemFontOfSize:15]];
    percGrowthLabel.textColor = [UIColor whiteColor];
    percGrowthLabel.backgroundColor = [UIColor clearColor];
    [percGrowthLabel setText:@"\% Growth"];

    // Stress
    UILabel *stressLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 390, 145,
21)];
    [stressLabel setFont:[UIFont boldSystemFontOfSize:15]];
    stressLabel.textColor = [UIColor whiteColor];
    stressLabel.backgroundColor = [UIColor clearColor];
    [stressLabel setText:@"Crop Stress"];

    // ETr
    UILabel *ETrLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 412, 145,
21)];
    [ETrLabel setFont:[UIFont boldSystemFontOfSize:15]];
    ETrLabel.textColor = [UIColor whiteColor];
    ETrLabel.backgroundColor = [UIColor clearColor];
    [ETrLabel setText:@"Ref ET"];

    // Adds labels that are filled by the array values
    // Station Location name
    UILabel *projectNameLabel  = [[UILabel alloc] initWithFrame:CGRectMake(40, 65,
260, 35)];
    [projectNameLabel setFont:[UIFont boldSystemFontOfSize:22]];
```

```
    projectNameLabel.textColor = [UIColor whiteColor];
    projectNameLabel.backgroundColor = [UIColor clearColor];
    [projectNameLabel setText:self.projectName];

    // Field Name
    UILabel *fieldNameLabel = [[UILabel alloc] initWithFrame:CGRectMake(40, 95,
275, 79)];
    [fieldNameLabel setFont:[UIFont systemFontOfSize:100]];// big font since it will size
down to fit label later on.
    fieldNameLabel.textColor = [UIColor whiteColor];
    fieldNameLabel.backgroundColor = [UIColor clearColor];
    [fieldNameLabel setText:self.fieldName];
    //resize font to fit label
    fieldNameLabel.numberOfLines = 1;
    fieldNameLabel.adjustsFontSizeToFitWidth = YES;
    float fontSize = fieldNameLabel.font.xHeight;
    [fieldNameLabel setFont:[UIFont systemFontOfSize:fontSize]];

    // Changes background color as a function of max temperature
    float number = [self.tempMax floatValue];

    if (number > 86) {
        // Red background if above values for GDD
        UIView *redView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 580)];
        redView.backgroundColor = [UIColor colorWithRed:.55 green:.2 blue:.25 alpha:.9];
        [self.view addSubview:redView];
    } else if (number >= 50) {
        // Green background if in range for GDD
        UIView *greenView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320,
580)];
        greenView.backgroundColor = [UIColor colorWithRed:.06666667 green:.2941177
blue:.2117647 alpha:1];
        [self.view addSubview:greenView];
    } else if (number >= 32) {
        // Blue background if below values for GDD
        UIView *blueView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320,
580)];
        blueView.backgroundColor = [UIColor colorWithRed:.04 green:.32 blue:.64
alpha:.8];
        [self.view addSubview:blueView];
    } else {
        // Gray background if below freezing
        UIView *grayView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320,
580)];
        grayView.backgroundColor = [UIColor colorWithRed:.3 green:.3 blue:.3 alpha:.9];
        [self.view addSubview:grayView];
```

```
    }

    // Yesterday ET Number
    UILabel *yesterdayETLabel = [[UILabel alloc] initWithFrame:CGRectMake(245, 195,
55, 21)];
    [yesterdayETLabel setFont:[UIFont boldSystemFontOfSize:17]];
    yesterdayETLabel.textColor = [UIColor whiteColor];
    yesterdayETLabel.backgroundColor = [UIColor clearColor];
    yesterdayETLabel.text = [NSString stringWithFormat:@"%.2f\"", [self.etc
floatValue]];

    // Precipitation number
    UILabel *precipNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 280,
140, 21)];
    [precipNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    precipNumLabel.textColor = [UIColor whiteColor];
    precipNumLabel.backgroundColor = [UIColor clearColor];
    precipNumLabel.textAlignment = NSTextAlignmentRight;
    precipNumLabel.text = [NSString stringWithFormat:@"%.2f\"", [self.precip
floatValue]];

    // Max Temp number
    UILabel *maxTempNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
302, 140, 21)];
    [maxTempNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    maxTempNumLabel.textColor = [UIColor whiteColor];
    maxTempNumLabel.backgroundColor = [UIColor clearColor];
    maxTempNumLabel.textAlignment = NSTextAlignmentRight;
    maxTempNumLabel.text = [NSString stringWithFormat:@"%dºF" , [self.tempMax
intValue]];

    // Min Temp number
    UILabel *minTempNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
324, 140, 21)];
    [minTempNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    minTempNumLabel.textColor = [UIColor whiteColor];
    minTempNumLabel.backgroundColor = [UIColor clearColor];
    minTempNumLabel.textAlignment = NSTextAlignmentRight;
    minTempNumLabel.text = [NSString stringWithFormat:@"%dºF", [self.tempMin
intValue]];

    // Total GDD number
    UILabel *totalGDDNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
346, 140, 21)];
    [totalGDDNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    totalGDDNumLabel.textColor = [UIColor whiteColor];
```

```
    totalGDDNumLabel.backgroundColor = [UIColor clearColor];
    totalGDDNumLabel.textAlignment = NSTextAlignmentRight;
    totalGDDNumLabel.text = [NSString stringWithFormat:@"%d", [self.gdd intValue]];

    // Total % Growth number
    UILabel *percGrowthNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
368, 140, 21)];
    [percGrowthNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    percGrowthNumLabel.textColor = [UIColor whiteColor];
    percGrowthNumLabel.backgroundColor = [UIColor clearColor];
    percGrowthNumLabel.textAlignment = NSTextAlignmentRight;
    //Need to change to Percent
    percGrowthNumLabel.text = [NSString stringWithFormat:@"%.1f", ([self.percGrowth
floatValue] *100)];

    // Stress, yes or no?
    UILabel *stressNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 390,
140, 21)];
    [stressNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    stressNumLabel.textColor = [UIColor whiteColor];
    stressNumLabel.backgroundColor = [UIColor clearColor];
    stressNumLabel.textAlignment = NSTextAlignmentRight;
    // Need to say yes or no
    // If ks is less than 1, crop is in stress
    float ksNum = [self.ks floatValue];
    if (ksNum == 1) {
        self.ks = @"No";
    } else {
        self.ks = @"Yes";
    }
    stressNumLabel.text = [NSString stringWithFormat:@"%@", self.ks];

    // ETr value
    UILabel *ETrValue = [[UILabel alloc] initWithFrame:CGRectMake(92, 412, 140,
21)];
    [ETrValue setFont:[UIFont boldSystemFontOfSize:15]];
    ETrValue.textColor = [UIColor whiteColor];
    ETrValue.backgroundColor = [UIColor clearColor];
    ETrValue.textAlignment = NSTextAlignmentRight;
    ETrValue.text = [NSString stringWithFormat:@"%.2f\"", ([self.etr floatValue])];

    // GDD bar with text inside
    UILabel *GDDBar = [[UILabel alloc] initWithFrame:CGRectMake(20,170,185,25)];
    [GDDBar setFont:[UIFont boldSystemFontOfSize:12]];
    GDDBar.textColor = [UIColor lightTextColor];
    GDDBar.backgroundColor = [UIColor colorWithRed:.06666667 green:.55
```

```
blue:.2117647 alpha:1];
    GDDBar.textAlignment = NSTextAlignmentCenter;
    [GDDBar setText:@"0% ---Percent Growth--- 100%"];
    // Black boarder line around GDD bar
    GDDBar.layer.borderColor = [UIColor blackColor].CGColor;
    GDDBar.layer.borderWidth = 1.0;
    GDDBar.layer.cornerRadius = 5;
    GDDBar.clipsToBounds = YES;

    // Get the percent growth and tell where to put on GDD bar
    float percentGrowthLine = [percGrowth floatValue] * 184 + 19;
    NSLog(@"per Growthline = %f, %f", percentGrowthLine, [percGrowth floatValue]);

    // Create line
    UIView *GDDLine = [[UIView alloc]
initWithFrame:CGRectMake(percentGrowthLine, 170, 2, 35)];
    GDDLine.backgroundColor = [UIColor blackColor];

    //Center the text bubble, 15 for half-width, 1 more for line size.
    float percentGrowthValueLine = percentGrowthLine - 14;

    // Place circle below line with value of percent GDD
    UILabel *GDDPercCircle = [[UILabel alloc]
initWithFrame:CGRectMake(percentGrowthValueLine, 205, 30, 30)];
    [GDDPercCircle setFont:[UIFont boldSystemFontOfSize:10]];
    GDDPercCircle.textColor = [UIColor blackColor];
    GDDPercCircle.backgroundColor = [UIColor lightTextColor];
    GDDPercCircle.textAlignment = NSTextAlignmentCenter;
    GDDPercCircle.layer.cornerRadius = 15.0;
    GDDPercCircle.layer.masksToBounds = YES;
    GDDPercCircle.text = [NSString stringWithFormat:@"%.0f%%", ([self.percGrowth
floatValue] *100)];

    // Add all labels to the scrollview
    [self.view addSubview:previousDayEtLabel];
    [self.view addSubview:backgroundLabel];
    [self.view addSubview:todayLabel];
    [self.view addSubview:precipLabel];
    [self.view addSubview:maxTempLabel];
    [self.view addSubview:minTempLabel];
    [self.view addSubview:totalGDDLabel];
    [self.view addSubview:percGrowthLabel];
    [self.view addSubview:stressLabel];
    [self.view addSubview:ETrLabel];

    [self.view addSubview:projectNameLabel];
```

```
    [self.view addSubview:fieldNameLabel];
    [self.view addSubview:yesterdayETLabel];
    [self.view addSubview:precipNumLabel];
    [self.view addSubview:maxTempNumLabel];
    [self.view addSubview:minTempNumLabel];
    [self.view addSubview:totalGDDNumLabel];
    [self.view addSubview:percGrowthNumLabel];
    [self.view addSubview:stressNumLabel];
    [self.view addSubview:ETrValue];

    [self.view addSubview:GDDBar];
    [self.view addSubview:GDDLine];
    [self.view addSubview:GDDPercCircle];

    // Enable paging of the scrollview
    //[self.scrollView setPagingEnabled:YES];
    //self.scrollView.showsHorizontalScrollIndicator = NO;

    //self.pageControl.numberOfPages = 1;//pageCount;
    //self.pageControl.currentPage = 0;
}

- (void)viewDidUnload
{
    [super viewDidUnload];
}

/*
// Activates the pageControl
- (void)scrollViewDidScroll:(UIScrollView *)scrollView {
    CGFloat pageWidth = self.scrollView.frame.size.width; // need iVar with getter for
scrollView
    float fractionalPage = self.scrollView.contentOffset.x / pageWidth;
    NSInteger page = lround(fractionalPage);
    self.pageControl.currentPage = page; // need iVar with getter for pageControl
}
*/
- (IBAction)doneWeather:(id)sender {
    [self dismissViewControllerAnimated:YES completion:NULL];
}
@end
```

```
//  WaterWeatheriPadView.h

#import <UIKit/UIKit.h>
```

```
@interface WaterWeatheriPadView : UIView

@end
```

```
//  WaterWeatheriPadView.m
#import "WaterWeatheriPadView.h"

@interface WaterWeatheriPadView ()
@property (nonatomic, strong) NSString *fieldName;
@property (nonatomic, strong) NSString *projectName;
@property (nonatomic, strong) NSString *dateWeather;
@property (nonatomic, strong) NSString *etc;
@property (nonatomic, strong) NSString *gdd;
@property (nonatomic, strong) NSString *etr;
@property (nonatomic, strong) NSString *precip;
@property (nonatomic, strong) NSString *tempMax;
@property (nonatomic, strong) NSString *tempMin;
@property (nonatomic, strong) NSString *percGrowth;
@property (nonatomic, strong) NSString *ks;

@end

@implementation WaterWeatheriPadView


@synthesize fieldName, projectName, dateWeather, etc, gdd, precip, tempMin,
tempMax, percGrowth, ks;

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code
    }
    return self;
}

// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance during animation.
- (void)drawRect:(CGRect)rect
{
    //Load saved variables
    self.fieldName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"fieldNameSaved"];
```

```
    self.projectName = [[NSUserDefaults standardUserDefaults]
stringForKey:@"projectNameSaved"];
    self.dateWeather = [[NSUserDefaults standardUserDefaults]
stringForKey:@"dateWeatherSaved"];
    self.etc = [[NSUserDefaults standardUserDefaults] stringForKey:@"etcSaved"];
    self.gdd = [[NSUserDefaults standardUserDefaults] stringForKey:@"gddSaved"];
    self.etr = [[NSUserDefaults standardUserDefaults] stringForKey:@"etrSaved"];
    self.precip = [[NSUserDefaults standardUserDefaults]
stringForKey:@"precipitationSaved"];
    self.tempMax = [[NSUserDefaults standardUserDefaults]
stringForKey:@"tempMaxSaved"];
    self.tempMin =[[NSUserDefaults standardUserDefaults]
stringForKey:@"tempMinSaved"];
    self.percGrowth =[[NSUserDefaults standardUserDefaults]
stringForKey:@"percGrowthSaved"];
    self.ks =[[NSUserDefaults standardUserDefaults] stringForKey:@"ksSaved"];

    // These labels are constant
    // Previous Day ET
    UILabel *previousDayEtLabel = [[UILabel alloc] initWithFrame:CGRectMake(160,
170, 128, 21)];
    [previousDayEtLabel setFont:[UIFont systemFontOfSize:17]];
    previousDayEtLabel.textColor = [UIColor whiteColor];
    previousDayEtLabel.backgroundColor = [UIColor clearColor];
    previousDayEtLabel.textAlignment = NSTextAlignmentRight;
    [previousDayEtLabel setText:@"Daily ET"];

    // Background box
    UILabel *backgroundLabel = [[UILabel alloc] initWithFrame:CGRectMake(78, 250,
164, 191)];
    backgroundLabel.backgroundColor = [UIColor colorWithRed:0 green:0 blue:0
alpha:.4];

    UILabel *todayLabel = [[UILabel alloc] initWithFrame:CGRectMake(78, 255, 164,
21)];
    [todayLabel setFont:[UIFont boldSystemFontOfSize:17]];
    todayLabel.textColor = [UIColor whiteColor];
    todayLabel.backgroundColor = [UIColor clearColor];
    todayLabel.textAlignment = NSTextAlignmentCenter;
    [todayLabel setText:self.dateWeather];

    // Precip
    UILabel *precipLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 280, 145,
21)];
    [precipLabel setFont:[UIFont boldSystemFontOfSize:15]];
    precipLabel.textColor = [UIColor whiteColor];
```

```
    precipLabel.backgroundColor = [UIColor clearColor];
    [precipLabel setText:@"Precip"];

    // Max Temp
    UILabel *maxTempLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 302,
145, 21)];
    [maxTempLabel setFont:[UIFont boldSystemFontOfSize:15]];
    maxTempLabel.textColor = [UIColor whiteColor];
    maxTempLabel.backgroundColor = [UIColor clearColor];
    [maxTempLabel setText:@"High"];

    // Min Temp
    UILabel *minTempLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 324,
145, 21)];
    [minTempLabel setFont:[UIFont boldSystemFontOfSize:15]];
    minTempLabel.textColor = [UIColor whiteColor];
    minTempLabel.backgroundColor = [UIColor clearColor];
    [minTempLabel setText:@"Low"];

    // Total GDD
    UILabel *totalGDDLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 346,
145, 21)];
    [totalGDDLabel setFont:[UIFont boldSystemFontOfSize:15]];
    totalGDDLabel.textColor = [UIColor whiteColor];
    totalGDDLabel.backgroundColor = [UIColor clearColor];
    [totalGDDLabel setText:@"Total GDD"];

    // Perc. Growth
    UILabel *percGrowthLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 368,
145, 21)];
    [percGrowthLabel setFont:[UIFont boldSystemFontOfSize:15]];
    percGrowthLabel.textColor = [UIColor whiteColor];
    percGrowthLabel.backgroundColor = [UIColor clearColor];
    [percGrowthLabel setText:@"\% Growth"];

    // Stress
    UILabel *stressLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 390, 145,
21)];
    [stressLabel setFont:[UIFont boldSystemFontOfSize:15]];
    stressLabel.textColor = [UIColor whiteColor];
    stressLabel.backgroundColor = [UIColor clearColor];
    [stressLabel setText:@"Crop Stress"];

    // ETr
    UILabel *ETrLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 412, 145,
21)];
```

```
    [ETrLabel setFont:[UIFont boldSystemFontOfSize:15]];
    ETrLabel.textColor = [UIColor whiteColor];
    ETrLabel.backgroundColor = [UIColor clearColor];
    [ETrLabel setText:@"Ref ET"];

    // Adds labels that are filled by the array values
    // Station Location name
    UILabel *projectNameLabel  = [[UILabel alloc] initWithFrame:CGRectMake(40, 65,
260, 35)];
    [projectNameLabel setFont:[UIFont boldSystemFontOfSize:22]];
    projectNameLabel.textColor = [UIColor whiteColor];
    projectNameLabel.backgroundColor = [UIColor clearColor];
    [projectNameLabel setText:self.projectName];

    // Field Name
    UILabel *fieldNameLabel = [[UILabel alloc] initWithFrame:CGRectMake(40, 95,
275, 79)];
    [fieldNameLabel setFont:[UIFont systemFontOfSize:100]];// big font since it will size
down to fit label later on.
    fieldNameLabel.textColor = [UIColor whiteColor];
    fieldNameLabel.backgroundColor = [UIColor clearColor];
    [fieldNameLabel setText:self.fieldName];
    //resize font to fit label
    fieldNameLabel.numberOfLines = 1;
    fieldNameLabel.adjustsFontSizeToFitWidth = YES;
    float fontSize = fieldNameLabel.font.xHeight;
    [fieldNameLabel setFont:[UIFont systemFontOfSize:fontSize]];

    // Green background
    UIView *greenView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 580)];
    greenView.backgroundColor = [UIColor colorWithRed:.06666667 green:.2941177
blue:.2117647 alpha:1];
    [self addSubview:greenView];

    // Yesterday ET Number
    UILabel *yesterdayETLabel = [[UILabel alloc] initWithFrame:CGRectMake(245, 195,
55, 21)];
    [yesterdayETLabel setFont:[UIFont boldSystemFontOfSize:17]];
    yesterdayETLabel.textColor = [UIColor whiteColor];
    yesterdayETLabel.backgroundColor = [UIColor clearColor];
    yesterdayETLabel.text = [NSString stringWithFormat:@"%.2f\"", [self.etc
floatValue]];

    // Precipitation number
    UILabel *precipNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 280,
140, 21)];
```

```
    [precipNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    precipNumLabel.textColor = [UIColor whiteColor];
    precipNumLabel.backgroundColor = [UIColor clearColor];
    precipNumLabel.textAlignment = NSTextAlignmentRight;
    precipNumLabel.text = [NSString stringWithFormat:@"%.2f\"", [self.precip
floatValue]];

    // Max Temp number
    UILabel *maxTempNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
302, 140, 21)];
    [maxTempNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    maxTempNumLabel.textColor = [UIColor whiteColor];
    maxTempNumLabel.backgroundColor = [UIColor clearColor];
    maxTempNumLabel.textAlignment = NSTextAlignmentRight;
    maxTempNumLabel.text = [NSString stringWithFormat:@"%dºF" , [self.tempMax
intValue]];

    // Min Temp number
    UILabel *minTempNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
324, 140, 21)];
    [minTempNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    minTempNumLabel.textColor = [UIColor whiteColor];
    minTempNumLabel.backgroundColor = [UIColor clearColor];
    minTempNumLabel.textAlignment = NSTextAlignmentRight;
    minTempNumLabel.text = [NSString stringWithFormat:@"%dºF", [self.tempMin
intValue]];

    // Total GDD number
    UILabel *totalGDDNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
346, 140, 21)];
    [totalGDDNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    totalGDDNumLabel.textColor = [UIColor whiteColor];
    totalGDDNumLabel.backgroundColor = [UIColor clearColor];
    totalGDDNumLabel.textAlignment = NSTextAlignmentRight;
    totalGDDNumLabel.text = [NSString stringWithFormat:@"%d", [self.gdd intValue]];

    // Total % Growth number
    UILabel *percGrowthNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92,
368, 140, 21)];
    [percGrowthNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    percGrowthNumLabel.textColor = [UIColor whiteColor];
    percGrowthNumLabel.backgroundColor = [UIColor clearColor];
    percGrowthNumLabel.textAlignment = NSTextAlignmentRight;
    //Need to change to Percent
    percGrowthNumLabel.text = [NSString stringWithFormat:@"%.1f", ([self.percGrowth
floatValue] *100)];
```

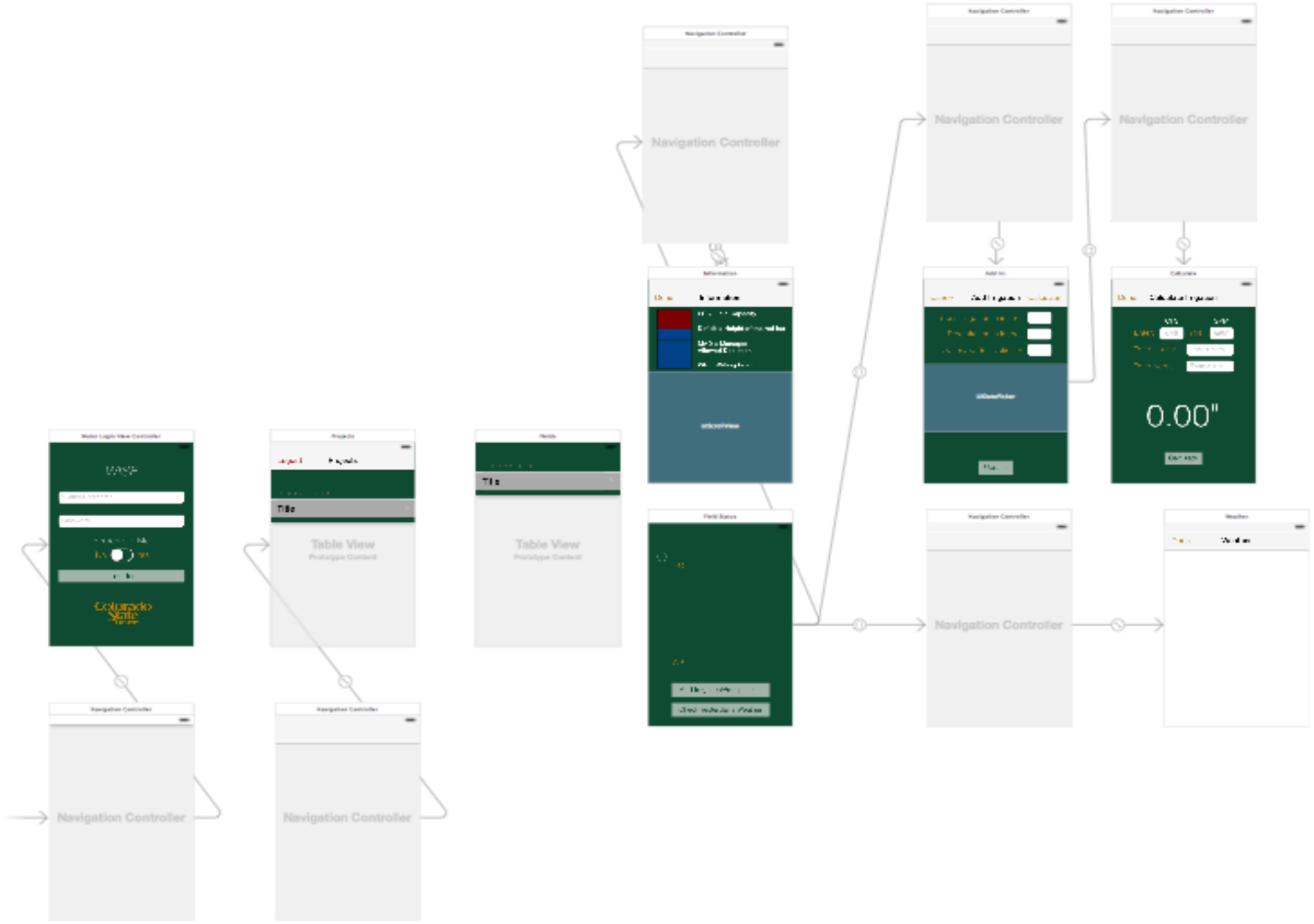
```
    // Stress, yes or no?
    UILabel *stressNumLabel = [[UILabel alloc] initWithFrame:CGRectMake(92, 390,
140, 21)];
    [stressNumLabel setFont:[UIFont boldSystemFontOfSize:15]];
    stressNumLabel.textColor = [UIColor whiteColor];
    stressNumLabel.backgroundColor = [UIColor clearColor];
    stressNumLabel.textAlignment = NSTextAlignmentRight;
    //Need to say yes or no
    // If ks is less than 1, crop is in stress
    float ksNum = [self.ks floatValue];
    if (ksNum == 1) {
        self.ks = @"No";
    } else {
        self.ks = @"Yes";
    }
    stressNumLabel.text = [NSString stringWithFormat:@"%@", self.ks];

    // ETr value
    UILabel *ETrValue = [[UILabel alloc] initWithFrame:CGRectMake(92, 412, 140,
21)];
    [ETrValue setFont:[UIFont boldSystemFontOfSize:15]];
    ETrValue.textColor = [UIColor whiteColor];
    ETrValue.backgroundColor = [UIColor clearColor];
    ETrValue.textAlignment = NSTextAlignmentRight;
    ETrValue.text = [NSString stringWithFormat:@"%.2f\"", ([self.etr floatValue])];

    // GDD bar with text inside
    UILabel *GDDBar = [[UILabel alloc] initWithFrame:CGRectMake(20,170,185,25)];
    [GDDBar setFont:[UIFont boldSystemFontOfSize:12]];
    GDDBar.textColor = [UIColor lightTextColor];
    GDDBar.backgroundColor = [UIColor colorWithRed:.06666667 green:.55
blue:.2117647 alpha:1];
    GDDBar.textAlignment = NSTextAlignmentCenter;
    [GDDBar setText:@"0% ---Percent Growth--- 100%"];
    // Black boarder line around GDD bar
    GDDBar.layer.borderColor = [UIColor blackColor].CGColor;
    GDDBar.layer.borderWidth = 1.0;
    GDDBar.layer.cornerRadius = 5;
    GDDBar.clipsToBounds = YES;

    // Get the percent growth and tell where to put on GDD bar
    float percentGrowthLine = [percGrowth floatValue] * 184 + 19;
    NSLog(@"per Growthline = %f, %f", percentGrowthLine, [percGrowth floatValue]);

    // Create line
```

```
    UIView *GDDLine = [[UIView alloc]
initWithFrame:CGRectMake(percentGrowthLine, 170, 2, 35)];
    GDDLine.backgroundColor = [UIColor blackColor];

    //Center the text bubble, 15 for half-width, 1 more for line size.
    float percentGrowthValueLine = percentGrowthLine - 14;

    // Place circle below line with value of percent GDD
    UILabel *GDDPercCircle = [[UILabel alloc]
initWithFrame:CGRectMake(percentGrowthValueLine, 205, 30, 30)];
    [GDDPercCircle setFont:[UIFont boldSystemFontOfSize:10]];
    GDDPercCircle.textColor = [UIColor blackColor];
    GDDPercCircle.backgroundColor = [UIColor lightTextColor];
    GDDPercCircle.textAlignment = NSTextAlignmentCenter;
    GDDPercCircle.layer.cornerRadius = 15.0;
    GDDPercCircle.layer.masksToBounds = YES;
    GDDPercCircle.text = [NSString stringWithFormat:@"%.0f%%", ([self.percGrowth
floatValue] *100)];

    // Add all labels to the scrollview
    [self addSubview:previousDayEtLabel];
    [self addSubview:backgroundLabel];
    [self addSubview:todayLabel];
    [self addSubview:precipLabel];
    [self addSubview:maxTempLabel];
    [self addSubview:minTempLabel];
    [self addSubview:totalGDDLabel];
    [self addSubview:percGrowthLabel];
    [self addSubview:stressLabel];
    [self addSubview:ETrLabel];

    [self addSubview:projectNameLabel];
    [self addSubview:fieldNameLabel];
    [self addSubview:yesterdayETLabel];
    [self addSubview:precipNumLabel];
    [self addSubview:maxTempNumLabel];
    [self addSubview:minTempNumLabel];
    [self addSubview:totalGDDNumLabel];
    [self addSubview:percGrowthNumLabel];
    [self addSubview:stressNumLabel];
    [self addSubview:ETrValue];

    [self addSubview:GDDBar];
    [self addSubview:GDDLine];
    [self addSubview:GDDPercCircle];
```

```
}

@end
```

---

iPhone Storyboard



iPad Storyboard