TESTING – PROCESS (ALGORITHM)

- Dynamic testing with various weather conditions
- Fast forward time for faster program testing
- Feed in dummy input from defined input set
  - User input stays same
  - Sensors
    - Temperature: Normal, High
    - Moisture: Low, Normal, High
  - Weather API
    - Chance of rain: None, % Chance of Rain
- Testing log – Write input and output to file to document test results

WEATHER FORECAST

- Parse XML data and assign to program variables

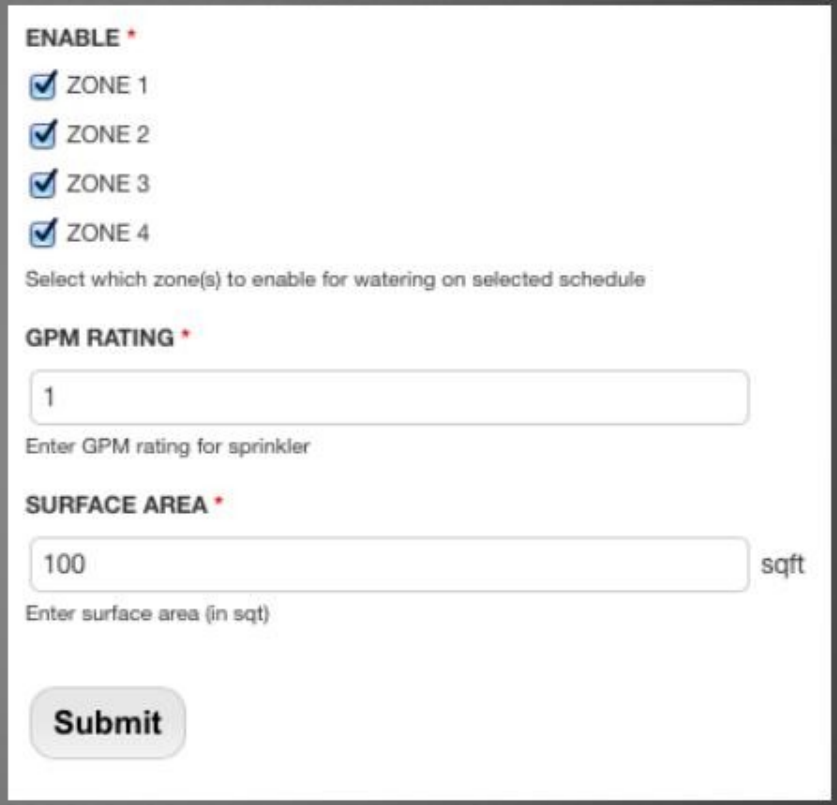
```
struct weatherData
{
   int dayOfWeek;        // day of the week
   int highTemp;         // high of the day
   int lowTemp;          // low of the day
   bool chanceOfRain;  // % chance of rain
   int humidity_avg;     // % of humidity
   float qpf_in;         // predicted rainfall in
   time_t epochTime;  // int time since epoch
   time_t updatedOn;  // last time updated
   string conditions;   // string check condition
};
```

MOTIVATION

- Many controllers today still not efficient enough
- Existing controllers do not take advantage of all information sources
- Existing controllers use sensors or weather data

USER DEFINED SETTINGS

- User defines additional settings
  - Zone selection
  - GPM rating of sprinklers
  - Surface area of zones
- Generates xml file on web server
  - After user clicks submit
  - Accessible by program on FX12



USER DEFINED SETTINGS

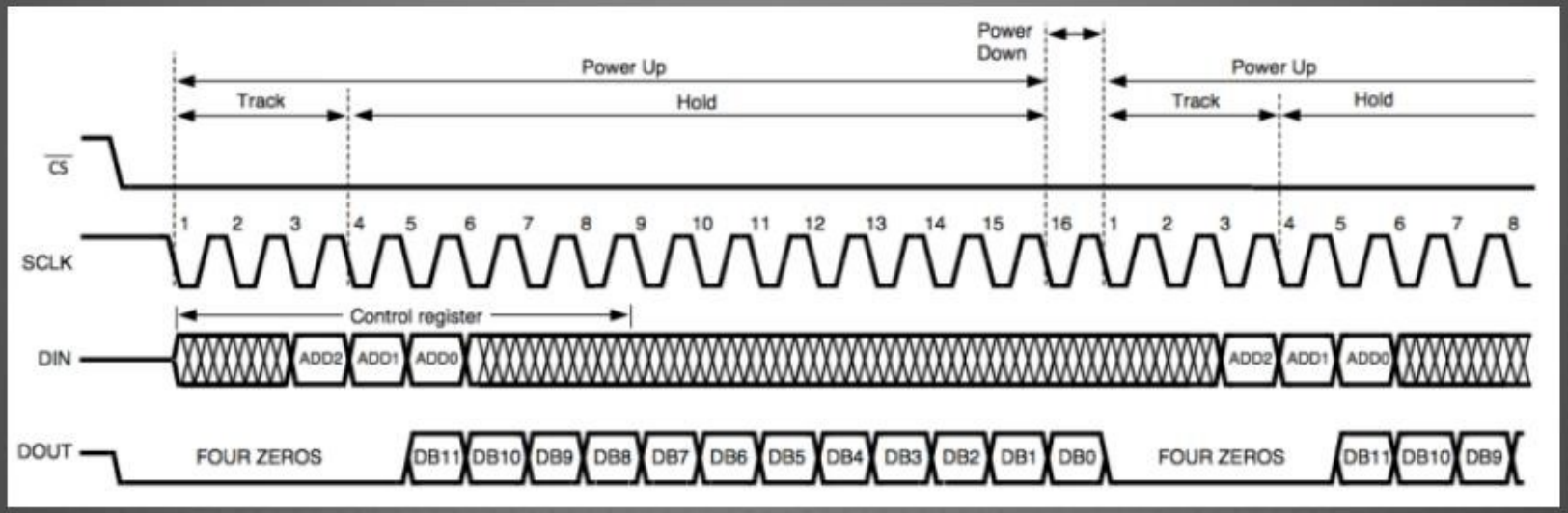
- Parse XML data and assign to program variables

```
struct userProgram
{
   string watering_days;  // int day Sun to Sat (0 - 6)
   string watering_time;  // user input as char hh:mm
   int duration;          // duration to water in min
   string watering_zone; // watering zone ex. 1,3
   int sprinklerGPM;      // sprinkler GPM rating
   int surfaceArea;       // zone surface area ft²
   time_t updatedOn;      // when data was updated
   time_t integerTime;    // conv time since epoch
};
```

```
▼<root>
   <zipcode>23284</zipcode>
   <watering_days>1,3</watering_days>
   <watering_time>14:30</watering_time>
   <enable>1,2,3,4</enable>
   <duration>30</duration>
   <gpm_rating>1</gpm_rating>
   <surface_area>100</surface_area>
   <updated>03/06/2014 12:24:23</updated>
 </root>
```

ADC

- Output 16 bits (last 12 bits hold converted voltage)



SYSTEM LOGIC

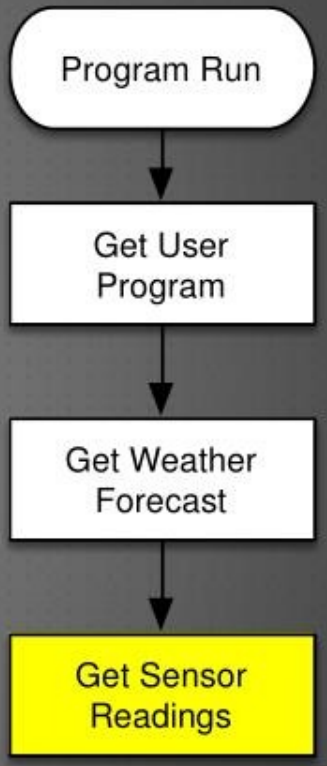
- Functions included in wateringAPI class

```
void getWeather();          // get weather forecast from wunderground
void getUserProgram();      // get user defined program from website
void getSensorValues();     // read sensors
real calc_water_output();   // calculates water volume in inches
void isStartTime();         // determine if SIS should start watering
void check_water_stop();    // determine if SIS should stop watering
void start_water();         // start the watering
void stop_water();          // stop the watering
int dayOfWeek(string);      // returns the day of week
int enumerate(string);      // converts string to integer
```

PROGRAM OPERATION

- Raw sensor values read
- Raw values converted to real values

```
void WateringAPI::getSensorValues()
{
    .
    .
    .
}
```



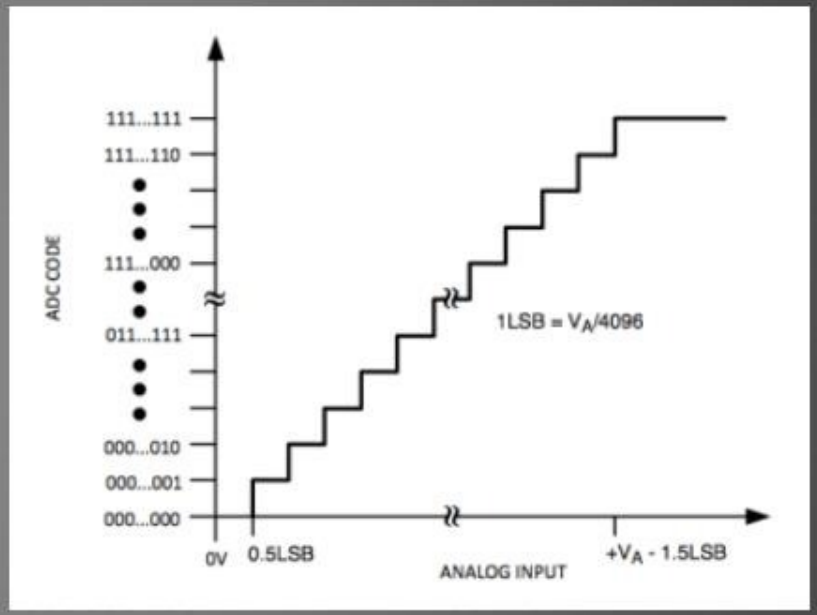
ORIGINAL BUDGET

10mm Green LED (COM-08861)	$1.50	Qty 4
Waterproof Temperature Sensor (DS18B20)	$9.95	Qty 1
Soil Moisture Sensor (SEN0114)	$4.80	Qty 1
NEMA-4 Enclosure (PN1341-C)	$37.00	Qty 1
4’ x 8’ Plywood for demo display	$35.00	Qty 1
Misc. wire, screws, standoffs	$10.00	
Signage	$25.00	

MOISTURE SENSOR

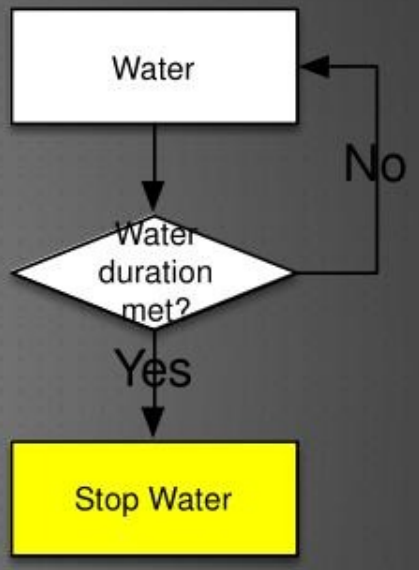
- Output voltage 0 – 4.2V
- Match scales
  - 0 – 300 <-> 0 – 1.4V : Dry Soil
  - 300 – 700 <-> 1.4 – 2.8V : Humid Soil
  - 700 – 950 <-> 2.8 – 4.2V : Wet Soil

- 0 value -> 0000 0000 0000
- 950 value -> 1101 0111 0000



PROGRAM OPERATION

- Stops the watering sequence
- Calculates the water run time and adds it to the totalizer for water runtime



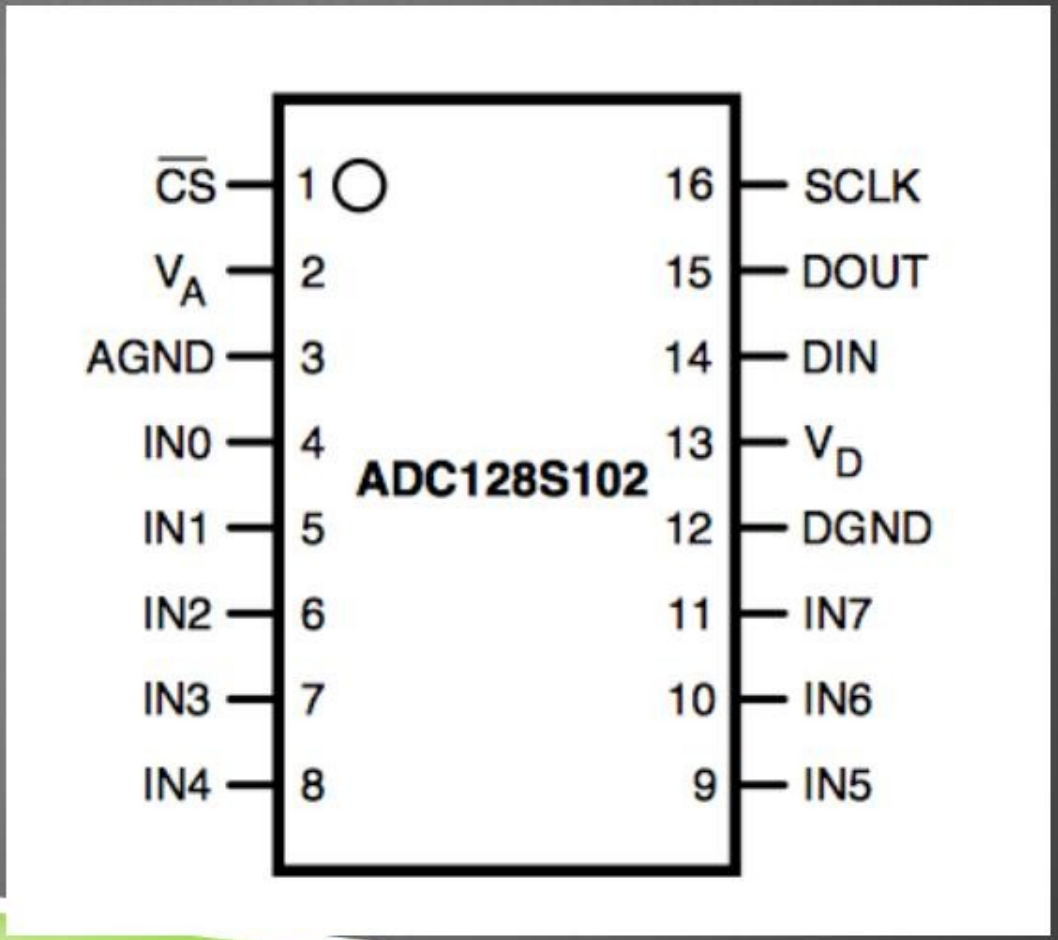

```
void WateringAPI::stop_water()
{
    .
    .
    .
}
```

INTRODUCTION

- Modern irrigation system
- Prediction and real-time input based system
- Deisgn goals:
  - Helps reduce waste
  - Prevent under watering
  - Prevent over watering
  - Easy to configure and operate

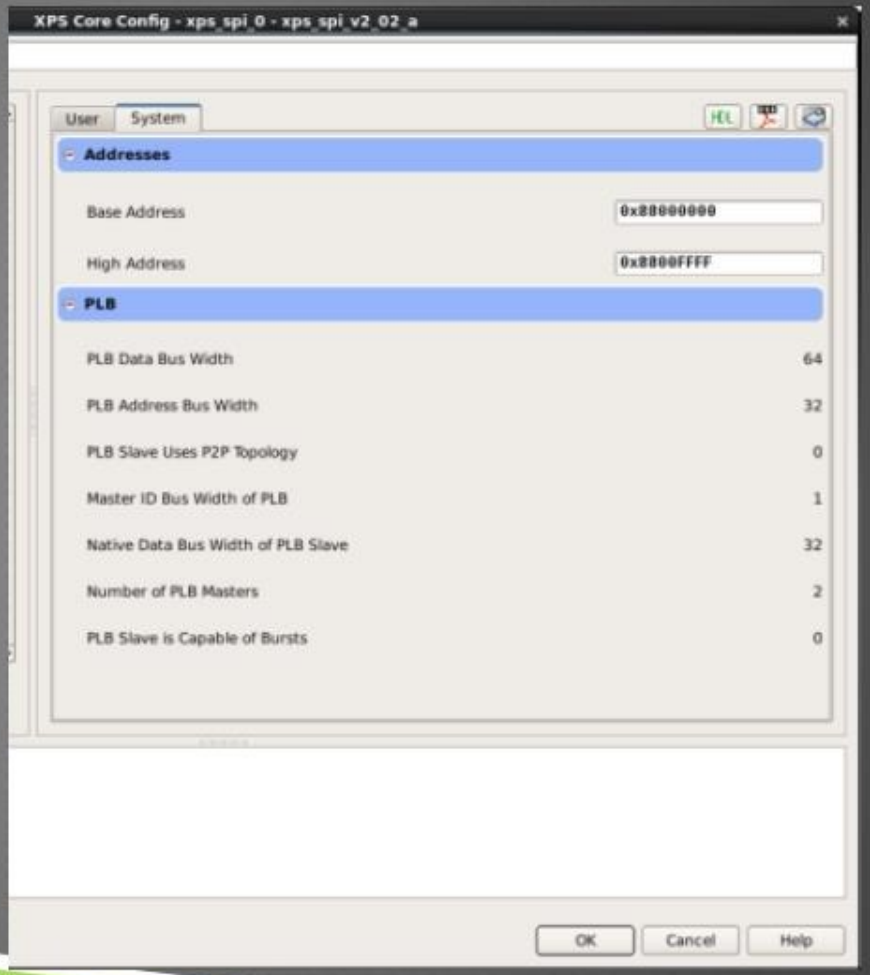
ADC

- Both sensors will use ADC through SPI
- IN0: Temperature
- IN1: Moisture



SPI INTERFACE

- SPI interface built
- Using 256 bytes



PROJECT STATUS

PROGRAM OPERATION

- Quick Water
  - At high temperatures above 95 degF
  - If no watering scheduled today
  - One-time water for half the user defined duration


```
void WateringAPI::isStartTime()
{
    .
    .
    .
}
```

DETAILED DESIGN
OUTPUTS

DETAILED DESIGN
INPUTS

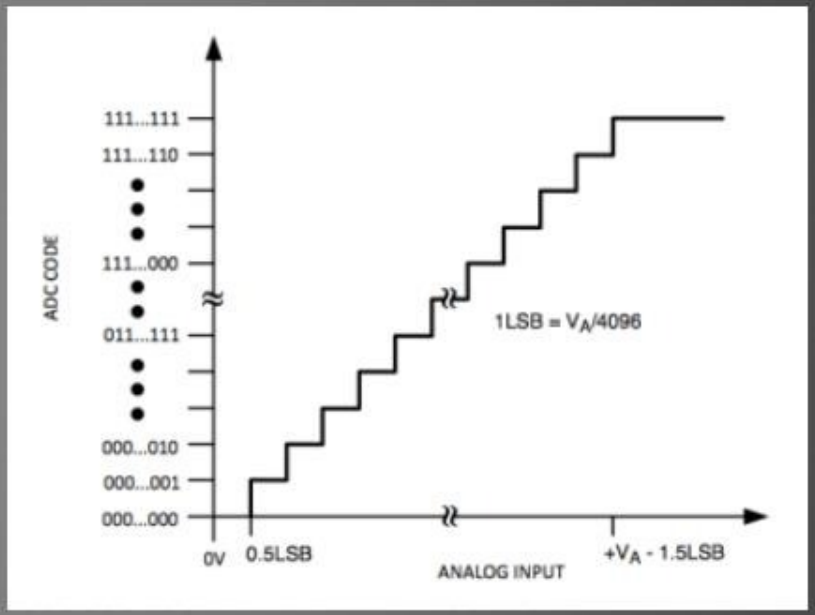
TASKS IN PROGRESS (ACTIVE)

- Software
  - Memory address mapping
    - SPI – ADC
  - Function implementation
    - getSensorValues()
    - calc_water_output()
    - isStartTime()
    - check_water_stop()

TEMPERATURE SENSOR

- Match scales
- 10 mV / degC
- Range 100mV to 2000mV
- 750mV at 25 degC

- -40 degC -> 0000 0000
- 125 degC -> 0110 0111

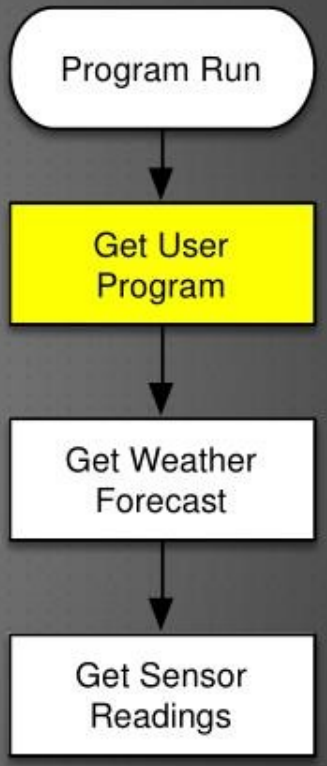


DETAILED DESIGN
PROCESS

PROGRAM OPERATION

- User defined settings and watering schedule fetched from website

```
void WateringAPI::getUserProgram()
{
    .
    .
    .
}
```



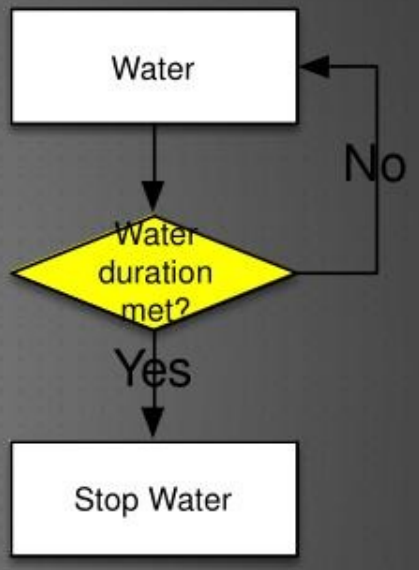
NTP CLIENT

- Install NTP package in Embedded Linux
- Configure NTP
  - server 0.rhel.pool.ntp.org
  - server 1.rhel.pool.ntp.org
  - server 2.rhel.pool.ntp.org
- Set service to run automatically
  - # /etc/init.d/ntpd start

DIVISION OF LABOR

PROGRAM OPERATION

- Determines if the user specified watering interval has occurred
- Checks if it has started raining




```
void WateringAPI::check_water_stop()
{
    .
    .
    .
}
```

NOT STARTED

- Procurement
  - Enclosure
  - Mounting hardware
- Hardware
  - LCD interface
  - UART
  - Digital Outputs
  - Synthesis
  - Construction
- Software
  - NTP client
  - Persistent IP address
  - File Output
  - LCD output
- Execute Test Plan
- System Documentation

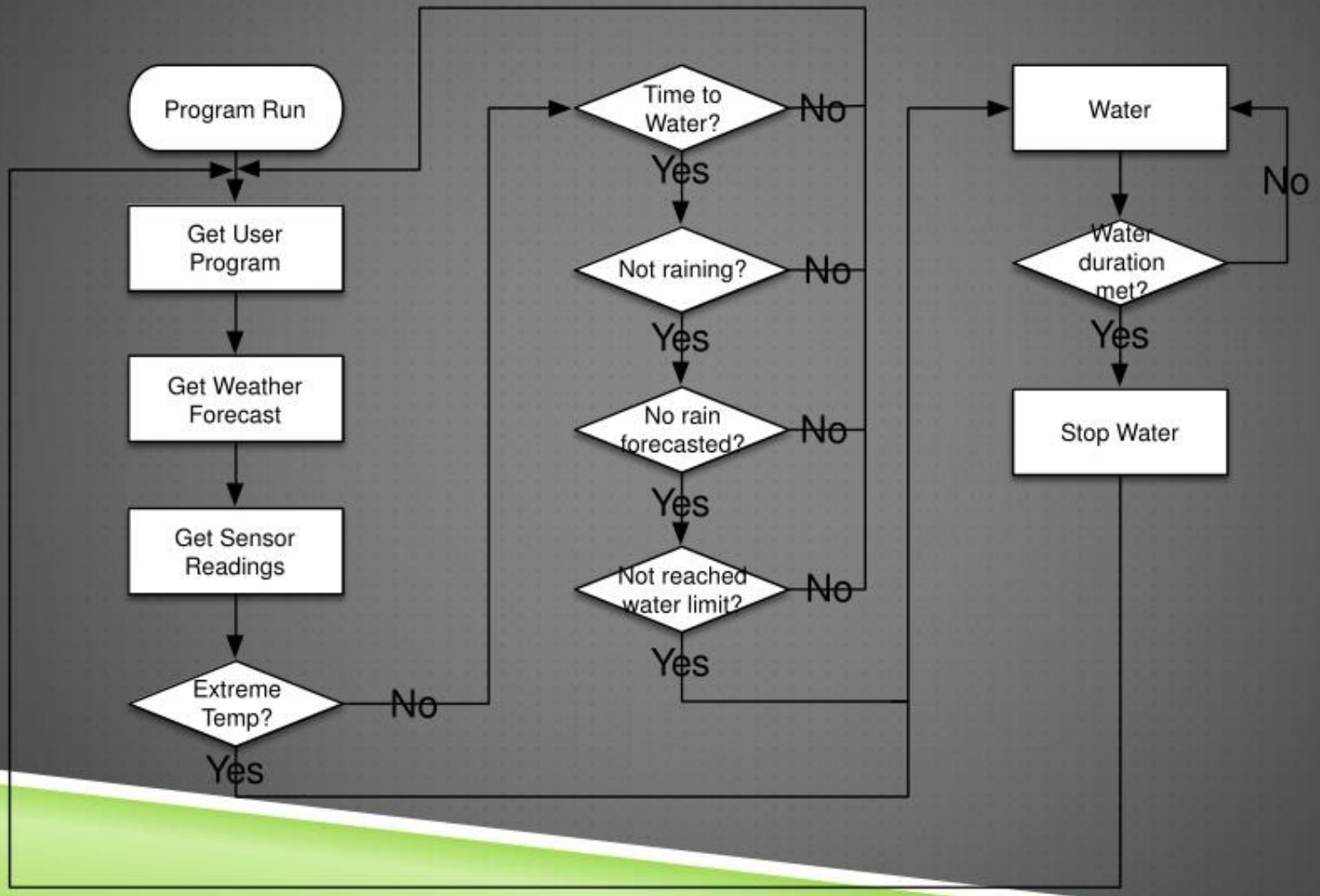
WORKFLOW

- The system workflow is broken into 3 categories
  - Input
  - Process
  - Output

MOISTURE SENSOR

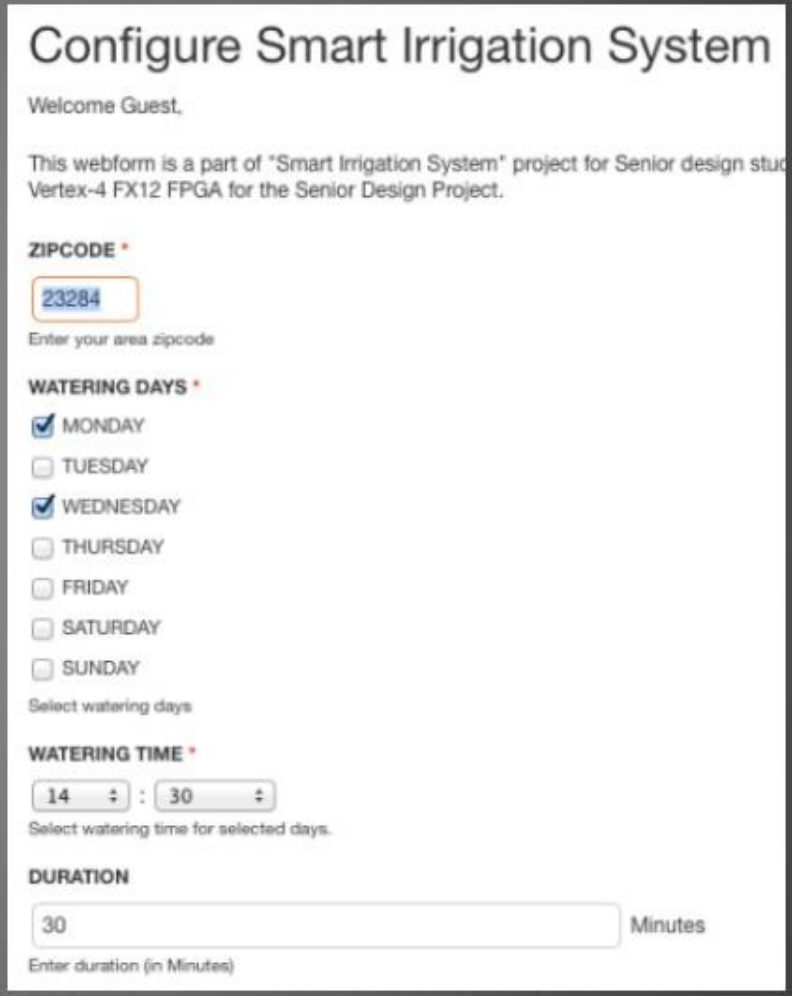
- Using SEN0114 moisture sensor
- Passes current through soil
  - Reads resistance to determine moisture
- Analog sensor selected
  - All moisture sensors found were analog
  - Inexpensive alternative to rain sensor
  - Not suitable for outdoor use as is
  - Place in soil where sprinklers do not water

SYSTEM PROCESS



USER DEFINED SETTINGS

- Web based form used
- External website user will access
  - Web server may not run on FX12 with limited space available
  - Easier to work with external web server
- User defines base watering schedule
  - Location, sprinkler and coverage area
  - Watering schedule



WEATHER FORECAST

- WeatherChannel (wunderground) API
  - Best option of three different API's considered (Yahoo, OpenWeatherMap.org)
  - Provides forecast including rain in inches
  - Provides history
- Send a fixed URL from the program
- XML response file generated and downloaded to board using wget

- system("wget http://api.wunderground.com/api/63ea9ff4ff38eafa/conditions/q/VA/Richmond.xml -O forecastrss.xml");

PROGRAM OPERATION

- Determine if watering should commence
- Requirements to start water
  - Day and time in user defined schedule has occurred
  - It is not raining
  - No rain is forecast for today
  - The weekly watering limit has not been reached

```
void WateringAPI::isStartTime()
{
    .
    .
    .
}
```

SYSTEM OUTPUT

- Sprinkler on or off

COMPARISON

Irrigation Controller	Program Schedule	Multiple Programs	Moisture Sensor	Temp Sensor	Weather Forecast	Weather Stats	PC Interface	Rainfall Totalizer	Multizone
The RainBird	Yes	Yes	Included	Included	No	Yes	No	No	12
Toro Evolution	Yes	Yes	Supported	No	No	No	No	No	4 to 16
The Irrigation Caddy	Yes	Yes	Supported	No	Requires PC	No	Yes	No	10
Cyber Rain	Yes	Yes	Supported	No	Requires PC	No	Yes	No	8 or 16
The Rain Machine	Yes	Yes	No	No	Built-in	Yes	Yes	Yes	12
HydraWise	Yes	Yes	Supported	No	Built-in	No	Yes	*Yes	16
HydroFlash	Yes	Yes	Supported	No	Built-in	No	Yes	No	16
Smart Irrigation System	Yes	Yes	Included	Included	Built-in	Yes	Yes	Yes	4

BUDGET

SMART IRRIGATION SYSTEM

Scott Francy

Gaurav Shah

VCU | Virginia Commonwealth University

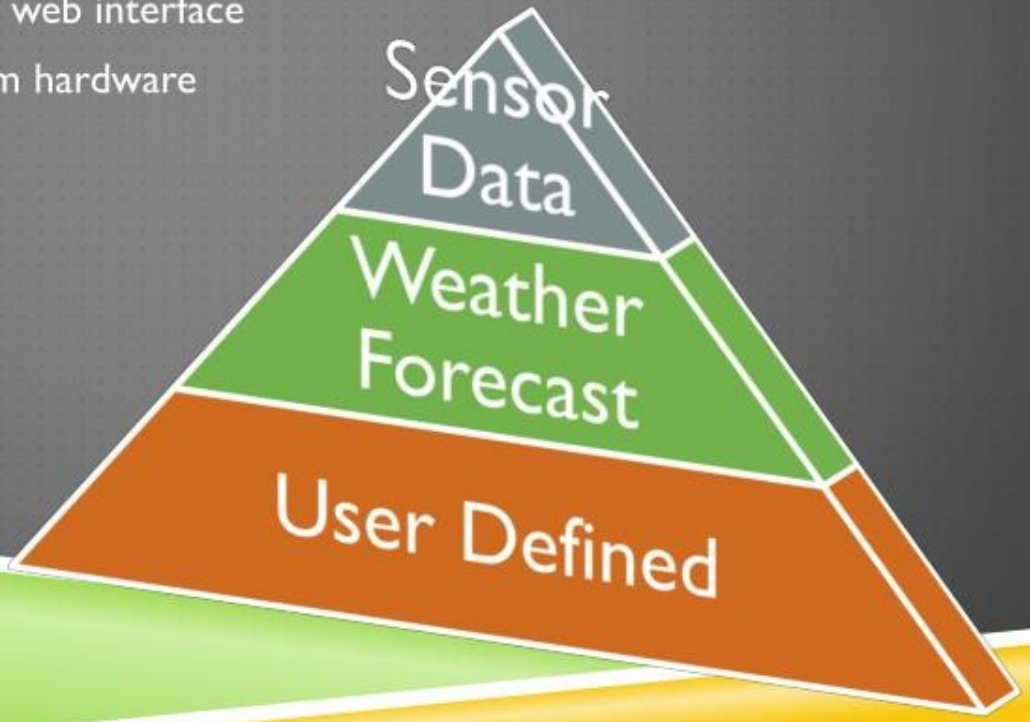
BACKGROUND

ENCLOSURE

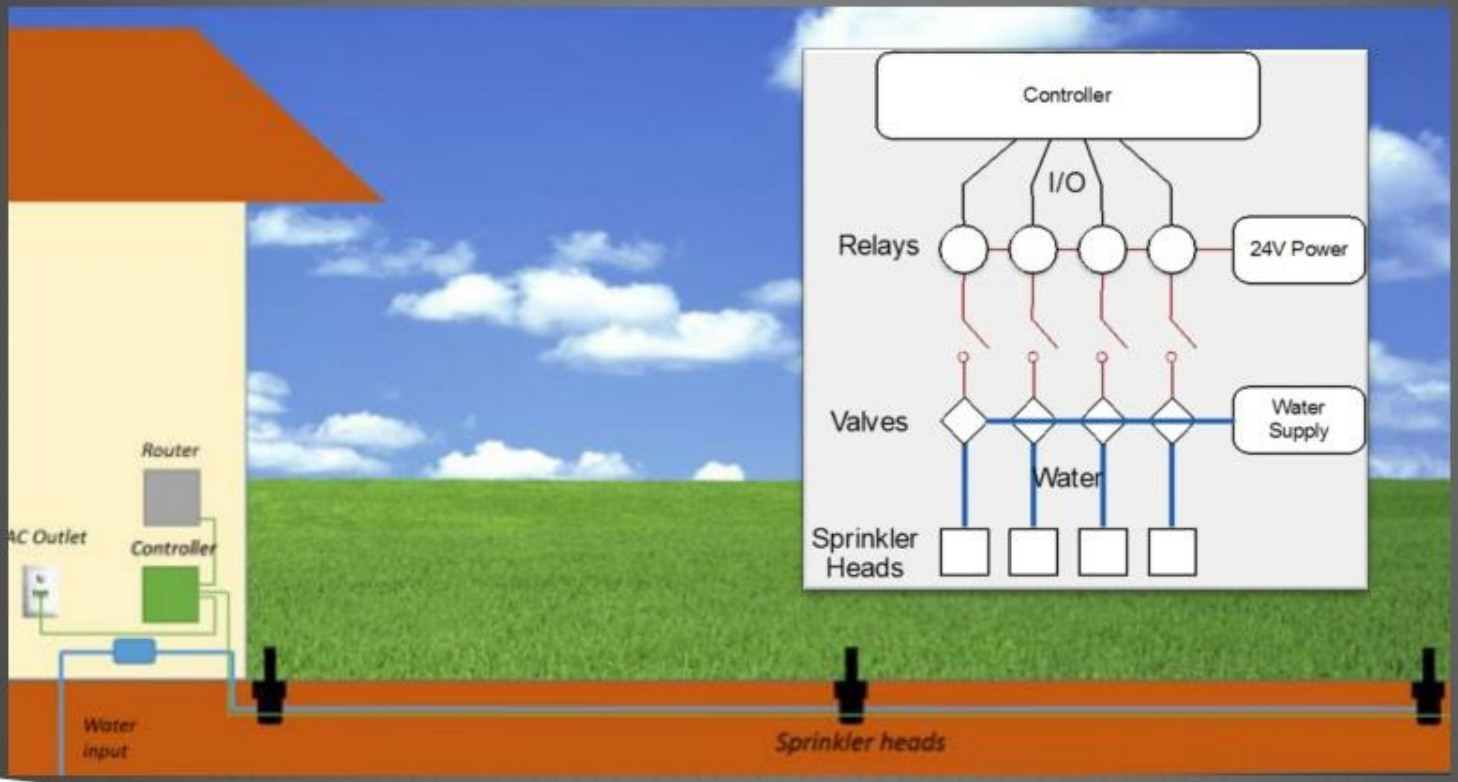
- NEMA-4 Enclosure (PN-1341-C)
  - Snug fit for reference board
  - Fix standoffs on bottom for ref board
  - Cutout for LCD
  - Holes for external inputs
    - Power
    - Moisture Sensor

SYSTEM INPUTS

- Embedded Linux
  - NTP client
- Program Input
  - Download weather data using Weather Underground API
  - Update user settings from web interface
  - Acquire sensor values from hardware
    - Temperature
    - Moisture



DESIGN


Controller
I/O
Relays
24V Power
Valves
Water Supply
Water
Sprinkler Heads
Router
AC Outlet
Controller
Water input
Sprinkler heads

PROGRAM OPERATION

- Total water volume is also calculated
- Two conflicting formulas found
  - 1) Water run time * GPM rating of sprinkler * 96.3 / sprinkler coverage area ft^2
  - 2) Water run time * GPM rating of sprinkler * .62 gallons / $1ft^2$
- Result is amount of water dispersed in inches
- This value is added to the downloaded rainfall volume

```
float WateringAPI::calc_water_output()
{
    .
    .
    .
}
```

SHARED TASKS

- Design Report
- CDR slides
- isStartTime() function
- check_water_stop() function
- CDR Report
- Test plan execution
- Documentation
- Final Design Report

TESTING - INPUTS

- Sensors
  - Collect samples at various temperature / moisture level
  - Check % Error
- WateringAPI Class
  - Write test code
  - Check individual components of the class, compare to expected output
  - Check data types (Structs)
    - Write data and read back
- User Input
  - Web based form validation
  - Data parsed properly

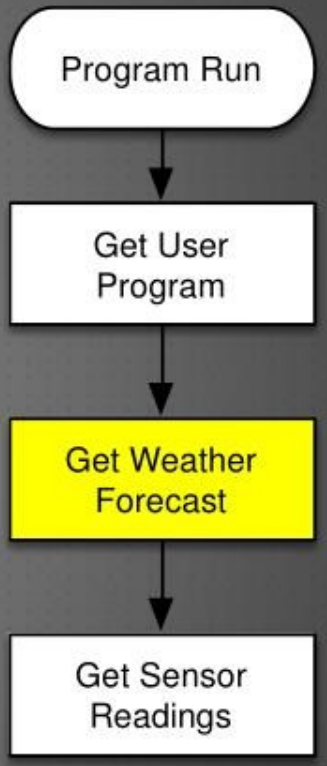
SCHEDULE CONT.

Task	Due Date
Refine program accuracy from test results	April 11
Execute full system test	April 11
Milestone: Testing and Analysis Complete	April 11
Build EXPO display	April 11
Senior Design EXPO	April 18
System Documentation	April 18
Build system demonstration program	April 25
Milestone: Implementation Complete	April 25
System Demonstration	April 29
Final Design Report	May 6

PROGRAM OPERATION

- Weather data fetched from wunderground
- XML file parsed
- Data stored into program variables

```
void WateringAPI::getWeather()
{
    .
    .
    .
}
```



SCOTT FRANCY

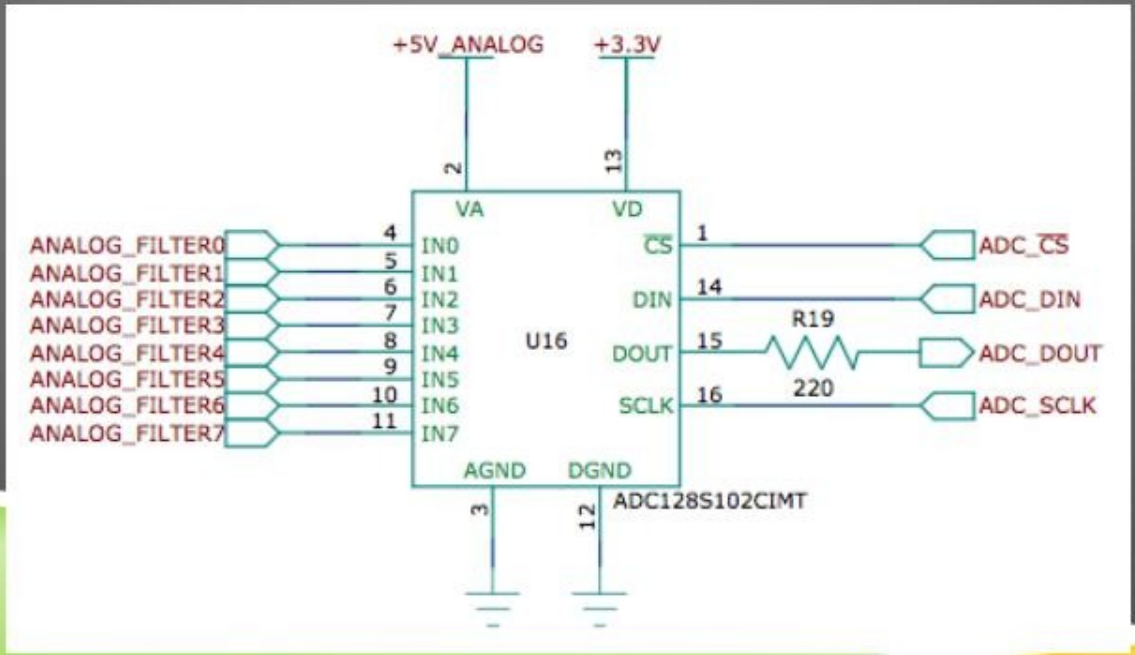
- Struct definitions
- getSensorValues() function
- calc_water_output() function
- start_water() function
- stop_water() function
- Procurement
- SPI – ADC
- Digital Outputs
- UART
- Synthesis
- LCD
- Construction

TASKS COMPLETED

- Procurement
  - Moisture sensor
  - Temperature sensor
  - LCD display
  - Wires
- Hardware
  - SPI build in EDK
  - Synthesis
- Software
  - Architecture, classes
  - Data types / structs
  - Function implementation
    - getWeather()
    - getUserProgram()
    - start_water()
    - stop_water()
    - dayOfWeek(string);
    - enumerate(string);

SENSORS

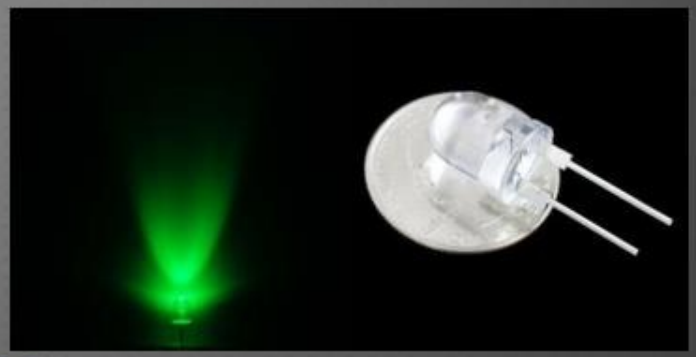
- Use SPI to interface with ADC
  - FX12 connected to ADC through reference board
  - ADC supports SPI



OUTPUT

- Output of system is turning the sprinklers on or off
- Controlled by start_water() and stop_water() functions
- Status maintained by bool isWatering variable

- LED's will be used to simulate sprinkler valve open / closed
  - 10mm Green LED (COM-08861)
  - Actual system would require relays and solenoid valves
  - Not required for prototype



OUTLINE

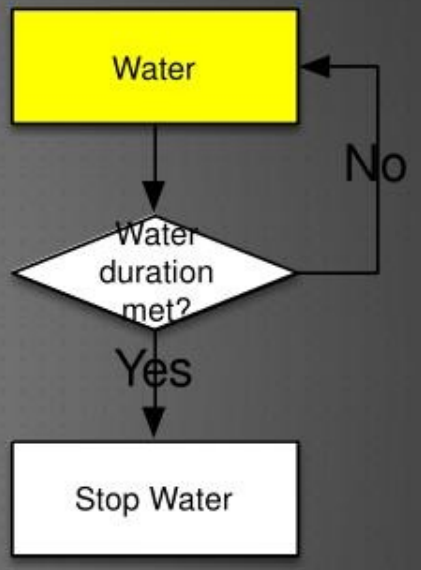
- Background
  - History
  - Motivation
- Design Summary
  - Overview
  - Workflow
- Detailed Design
  - Input
  - Process
  - Output
- Project Status
  - Tasks Completed
  - Tasks in Progress
  - Tasks not Started
- Risks
- Budget
- Divison of Labor
- Questions

OUTPUT

- Define GPIO, map pin locations in UCF
  - BUF_OUTPUT0 -> DO Header Pin 18
  - BUF_OUTPUT1 -> DO Header Pin 17
  - BUF_OUTPUT2 -> DO Header Pin 16
  - BUF_OUTPUT3 -> DO Header Pin 15
- Map virtual memory addresses to digital output hardware address
- Write to virtual memory address

PROGRAM OPERATION

- Starts the watering sequence




```
void WateringAPI::start_water()
{
    .
    .
    .
}
```

TESTING - OUTPUTS

- Verify output pins high when watering is active
  - Use LED's
- Verify LCD screen displaying correct data
  - Compare to expected output during testing
- Vary rain volume for totalizer validation
  - Compare rainfall totalizer to manually calculated weather history
  - Manually compare runtime * GPM * area formula to totalizer

LATEST BUDGET

10mm Green LED (COM-08861)	$1.50	Qty 4
Temperature Sensor (SEN-10988)	$1.50	Qty 2
Soil Moisture Sensor (SEN0114)	$5.95	Qty 1
Wires (40 count)	$5.63	Qty 1
LCD (LCD-09067)	$24.95	Qty 1
NEMA-4 Enclosure (PN1341-C)	$37.00	Qty 1
4' x 8' Plywood for demo display	$24.12	Qty 1
Misc. wire, screws, standoffs	$10.00	
Signage	$25.00	

RISKS

- NTP Client
  - Known to work on Linux
  - Unknown if any problems will exist with this embedded Linux image
  - Alternative is to use a time API
    - http://www.timeapi.org
- Rain Volume Calculation
  - Two conflicting formulas found to calculate watering volume
  - Need to research further

LCD DISPLAY

- LCD Display
  - 16x2
  - Serial Interface
  - 9600 to 38400 baud
  - 8-N-1
- Used to show current system status
  - Active / Inactive
  - While watering
    - Zone indication
    - Time Left
  - While idle
    - Next water time
    - Total water dispersed over last 7 days

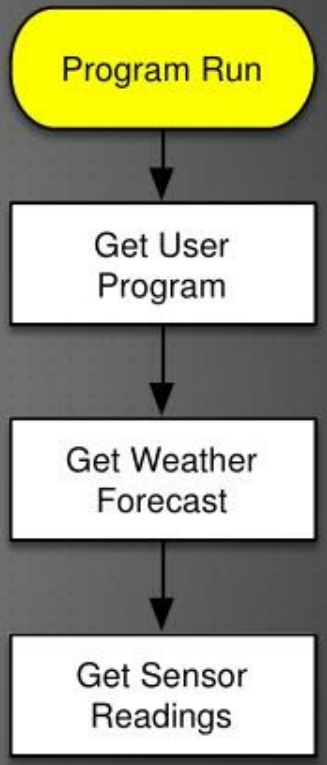
Q&A

HISTORY

- Water timers
- Water controller
- Water controller with advanced programming, user interface
- Internet enabled controller

PROGRAM OPERATION

- On FX12 power-on, Linux image loads
- NTP client synchronizes with pool.ntp.org
- Launches weather program



PLATFORM

- Xilinx Virtex-4 FX12 Mini Module with VCU Reference Board
  - FPGA
  - Xilinx PowerPC
  - Ethernet Interface
  - Flash Memory
  - 76 user defined I/O pins
  - On-board clock

DESIGN SUMMARY

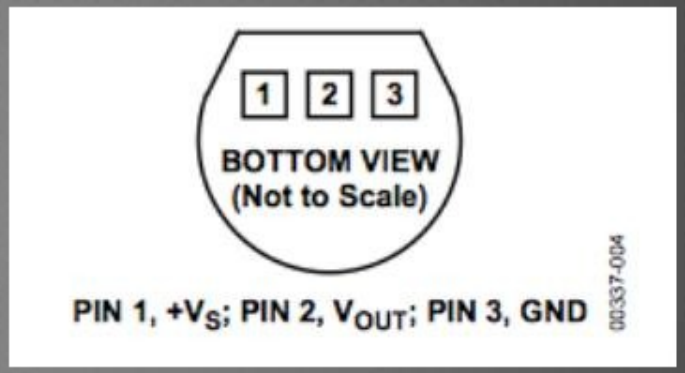
GAURAV SHAH

- Class definitions
- User Web Interface
- XML Parsing
- getWeather() function
- getUserProgram() function
- dayOfWeek() function
- Enumerate() function
- NTP client
- Persistent IP address
- Persistent file storage
- Test Plan

TEMPERATURE SENSOR

- Using TMP36 sensor
- Output is in degC
- Convert to degF
- -40 degC to 125 degC

- Analog sensor selected for simplicity
  - Other digital sensors proved problematic
  - Proprietary interfaces
  - 1-wire interfaces



OUTPUT

- Rolling 7 day totalizer of watering volume in inches
- If system resets, water volume tracking would be lost
- Save calculated values to file after every update
- Use persistent storage in flash (newfs.bin)
- Save value for each day to maintain rolling 7 day totalizer

RISKS

SCHEDULE

Task	Due Date
Fix SPI – ADC address mapping problem	March 14
Complete function implementation	March 14
Build UART and DO in EDK	March 14
NTP client	March 21
Output to LCD	March 21
Milestone: Hardware Interfaces Complete	March 21
Order enclosure and mounting hardware	March 21
Persistent IP address	March 28
File Output	March 28
Milestone: Initial Implementation Complete	April 4
Develop test program, demo mode	April 4
Execute full system test	April 4
Construction	April 4

TEMPERATURE SENSOR

- Sensor will be inside enclosure
- Need to compensate for additional heat inside enclosure
- Determine outside ambient temperature

SOFTWARE INTERFACE

- Sensors are checked continuously
- 12-bit binary value converted to real value
- Not critical to system functionality but critical to efficiency

```
struct fieldData
{
   int rawMoisture;       // reading is an int representing voltage
   float moistureValue;   // converted moisture value
   int rawTemp;           // reading is an integer representing voltage
   float tempValue;       // converted temperature value
   bool moistureHigh;     // determines whether ground is wet
};
```