NSF SmartFarm Project Report

NSF SmartFarm Meeting ___Date___
California State University Fresno, CA

PIC collage
Wireless Program Boards on Wake-up
SmartFarm Library for plug and play/IDE/Serial Screenshot
Raspberry Pi base station receive and push data to cloud for data analysis
  raspberry pi with xbee radio + cloud data screenshot
web IDE

Bo Liu
Nathan Oto
Caleb Fink
Table of Contents

I. Introduction ................................................................................................................................. 6

II. User Guide .................................................................................................................................. 6
   a. Plug and Play Measurement Data Collection ........................................................................ 6
      Required Materials: .................................................................................................................. 6
      Board Configuration: ................................................................................................................. 6
      SmartFarmIDE Configuration: ................................................................................................. 8
   b. SmartFarm Base Station Measurement and Uploading .......................................................... 9
      Required Materials.................................................................................................................. 10
      Uploading on Next Wake Cycle .............................................................................................. 10
      Posting Data on Cloud data host ThingSpeak ........................................................................ 10

III. Plug and Play Library for the SmartFarmIDE .......................................................................... 10
   a. Overview ................................................................................................................................. 10
   b. Approach ................................................................................................................................. 11
   c. Hardware .................................................................................................................................. 11
      Decagon 5TE Electrical Conductivity Sensor .......................................................................... 11
      DallasTemperature Waterproof Temperature Sensor ............................................................ 12
      Watermark Soil Moisture Sensor ............................................................................................. 12
   d. Software .................................................................................................................................... 13
      High Level User Functions ....................................................................................................... 13
      Mid Level User Functions ........................................................................................................ 13
      Low Level Functions ................................................................................................................ 15

IV. Raspberry Pi Smart Farm Base Station .................................................................................. 16
   a. General configuration of Raspberry Pi Model B ................................................................... 17
      Software and downloading the OS ........................................................................................... 17
      Guide to setting up the pi for the first time .............................................................................. 17
      Connecting the Pi to the internet ............................................................................................. 17
      Running the Raspberry Pi headless (without monitor, keyboard, or mouse) .......................... 18
      Giving the Raspberry Pi a Static IP Address .......................................................................... 19
   b. General configuration of SmartFarm Base Station ................................................................. 20
      PlatformIO Core 3.4 Installation and Uploading ..................................................................... 23
      Upload Test to the SmartFarm board from the Raspberry Pi Base Station ......................... 26
Configuration of SmartFarm Base Station as a WordPress Web Server .................................................. 27
Uploading a File from the WordPress Site to the Base Station .......................................................... 32
c. Wirelessly Programming a SmartFarm Board on the Next Wake Cycle ...................................... 36
d. Collecting SmartFarm Data and Posting to the Web ...................................................................... 36
   Sample SmartFarm Data Posted to ThingSpeak .............................................................................. 37
V. Overall Improvements ...................................................................................................................... 38
List of Figures

Figure 1. SmartFarm board v5.4 with Decagon sensor attached ............................................. 7
Figure 2. SmartFarm board v5.4 with DS18B20 Temperature Sensor attached ............................. 7
Figure 3. SmartFarm board v5.4 with WaterMark sensor attached ......................................... 8
Figure 4. SmartFarm board v5.4 with all three types of sensors attached .................................... 8
Figure 5. Ideal sketch .............................................................................................................. 9
Figure 6. Serial Printing of Measurement Data ....................................................................... 9
Figure 7. Updating the Raspberry Pi: sudo apt-get update ...................................................... 17
Figure 8. Upgrading the Raspberry Pi: sudo apt-get upgrade .................................................. 17
Figure 9. Headless Raspberry Pi .......................................................................................... 18
Figure 10. The PuTTY icon ................................................................................................. 18
Figure 11. Using the Raspberry Pi Hostname to connect via Putty ........................................... 19
Figure 12. Editing the dhcpcd.conf file for Static IP ............................................................... 19
Figure 13. dhcpcd.conf in nano editor with ethernet eth0 and wifi wlan0 ............................... 20
Figure 14. Command to check for the static IP address | ifconfig ........................................... 20
Figure 15. The WinSCP icon and download at https://winscp.net/eng/download.php ............... 21
Figure 16. Screenshot of using WinSCP .............................................................................. 21
Figure 17. Showing hidden files in WinSCP: Options > Preferences ...................................... 21
Figure 18. Showing hidden files in WinSCP: Panels, Check Show hidden files box and Click OK .... 22
Figure 19. The Notepad++ icon and download at https://notepad-plus-plus.org/download/v7.3.3.html ........................................................................................................................................... 22
Figure 20. Screenshot of using Notepad++ to edit python script files ....................................... 22
Figure 21. get-platformio.py script download ....................................................................... 23
Figure 22. Transfer file from computer to Raspberry Pi using WinSCP .................................. 23
Figure 23. Run the Command to install PlatformIO Core ....................................................... 23
Figure 24. The /home/pi/.platformio/packages/framework-arduinoavr directory contents ....... 24
Figure 25. Copying the variants, cores, and boards.txt into the /home/pi/.platformio/packages/framework-arduinoavr directory ................................................................. 24
Figure 26. PlatformIO boards recognition command ............................................................... 25
Figure 27. The SmartFarm board is recognized and ready to be initialized ............................... 25
Figure 28. Create the directory for a new PlatformIO project .................................................. 25
Figure 29. Change into the PlatformIO project directory just created ..................................... 25
Figure 30. Initialize the PlatformIO project for smartfarm board ............................................ 25
Figure 31. Run ls command in the lib directory to confirm the libraries needed to upload to SmartFarm ................................................................................................. 25
Figure 32. Compile the sketch in the src directory of the PlatformIO project .............................. 26
Figure 33. Upload the sketch in the src directory of the PlatformIO project to the SmartFarm v5.4 board .............................................................................................................. 26
Figure 34. Raspberry Pi is receiving serial data from the SmartFarm board with no sensors connected .... 27
Figure 35. Simple serial data python script Arduino_serial.py ............................................. 27
Figure 36. Install python-serial: sudo apt-get install python-serial ........................................... 27
Figure 37. Install apache2 package ....................................................................................... 28
Figure 38. Find your Raspberry Pi’s IP address ..................................................................... 28
Figure 39. The Apache2 Debian Default Page ....................................................................... 28
Figure 40. Install php command ........................................................................................... 28
Figure 41. Change the working directory to that of the server .......................................................... 28
Figure 42. Command to create index.php in nano .......................................................................... 28
Figure 43. Remove the index.html file ............................................................................................. 28
Figure 44. Restart the apache2 service if the index.php doesn’t show ............................................ 28
Figure 45. Installing MySQL and php-mysql packages ................................................................... 29
Figure 46. Remove all contents in the /var/www/html directory before downloading WordPress ...... 29
Figure 47. Download WordPress ...................................................................................................... 29
Figure 48. Extract the downloaded tarball .......................................................................................... 29
Figure 49. Move the folder to the current working directory .............................................................. 29
Figure 50. Remove the now empty folder to cleanup ......................................................................... 29
Figure 51. Change ownership of the wordpress files to the Apache user ........................................... 29
Figure 52. Start a mysql database .................................................................................................... 29
Figure 53. Creating the mysql database for Wordpress site ............................................................... 29
Figure 54. WordPress Welcome page ................................................................................................ 30
Figure 55. WordPress Dashboard, Settings, Permalinks ................................................................... 31
Figure 56. Permalinks, Settings, Post name setting, save the changes .............................................. 31
Figure 57. Apache’s rewrite mod enabled command ......................................................................... 31
Figure 58. Edit the Apache configuration file in the nano editor ....................................................... 32
Figure 59. Lines edited in Apache configuration file ......................................................................... 32
Figure 60. Install the WordPress File Upload plugin from the Add New under Plugins .................. 32
Figure 61. Change the ownership of the directory to include the new directory ............................... 33
Figure 62. Navigate to the lib folder of the WordPress File Upload plugin ....................................... 33
Figure 63. List the contents of lib and find the wfu_security.php file ............................................. 33
Figure 64. Locating the ino in the blacklist press Ctrl-W ................................................................. 33
Figure 65. Delete the entry once found, from the first comma to the 1. Ex: ...,”inl” => 1,”ino” => 1, “inp” => 1, goes to ...,”inl” => 1,”inp” => 1, ... ................................................................. 34
Figure 66. Click on an element to add a new page, Name the page, Save and Publish, then Close ........ 34
Figure 67. Add the Plugin Instance to the newly created Page .......................................................... 35
Figure 68. Editing to add a password protected Upload and Configuration .................................. 35
Figure 69. Upload File on the WordPress site .................................................................................... 36
Figure 70. The ThingSpeak Trademark by MathWorks ................................................................... 37
Figure 71. Sample SmartFarm WaterMark Data posted on ThingSpeak .......................................... 37
I. Introduction

SmartFarm Cal Poly team has taken on implementing a program for the v5.4 board that is plug and play with the sensors. The user plugs in the sensors and gets measurements without user setup, other than plugging them in. The team also has developed a Base Station to post data to the web, as well as remotely program the boards towards the future goal of a WebIDE.

Accomplishments:

1. Develop a SmartFarm library to make the system plug and play
   → The new SmartFarm Library makes the system user friendly by enabling the user to plug in sensors and get readings
2. Develop Raspberry Pi based base station and receive and push data to cloud for further data analysis.
   → Raspberry Pi Base Station developed to receive and send data to cloud via ThingSpeak for analysis
3. Wireless Program Circuit Boards at any time
   → Wireless Program Circuit Boards on Next Wake Cycle. XBee radio sleep mode doesn’t allow wakeup via over the air. Battery usage must be considered if wanting programming access at any time.

II. User Guide

a. Plug and Play Measurement Data Collection

The steps below describe the detailed steps for setting up and using the SmartFarm v5.4 board with the SmartFarmMeasure library. This is the basic setup, and can be altered as needed.

Required Materials:

1. SmartFarm v5.4 board
2. 3.7v LiPO Battery
3. Decagon 5TE EC sensor
4. DallasTemperature DS18B20 Temperature Sensor
5. WaterMark soil moisture sensor
6. XBee Wireless Programming Pair
7. Computer with SmartFarmIDE installed

Board Configuration:

1. Attach the Decagon 5TE sensor to one of the associated pin layouts
   a. The Decagon should have 3 wires in pigtail fashion
   b. The stripped wire represents ground, and should be connected to the socket marked by a ‘-‘
   c. The red wire represents the data bus and should be connected to the socket marked by a ‘s’
   d. The white wire represents the power source and should be connected to the socket marked by a ‘+’
e. The image below summarizes what an attached Decagon sensor should look like:

![Image of Decagon sensor attached](image1.png)

*Figure 1. SmartFarm board v5.4 with Decagon sensor attached*

2. Attach the DallasTemperature DS18B20 Temperature sensor to one of the associated pin layouts
   a. The DS18B20 should have 3 wires in pigtail fashion
   b. The black wire represents ground, and should be attached to the socket marked by a ‘-‘
   c. The green wire represents the data bus, and should be attached to the socket marked by a ‘s’
   d. The red wire represents power, and should be attached to the socket marked by a ‘+’
   e. The image below summarizes what an attached DS18B20 should look like:

![Image of DS18B20 Temperature sensor attached](image2.png)

*Figure 2. SmartFarm board v5.4 with DS18B20 Temperature Sensor attached*

3. Attach the WaterMark soil moisture sensors to one of the associated pin layouts
   a. The WaterMark sensor should have 2 identical wires
   b. Attach one wire to one side of an analog pin marked by ‘AN’, where N is the number 0-3
   c. Attach the other wire to the other side of the same analog pin
d. The image below summarizes what an attached WaterMark should look like:

![SmartFarm board v5.4 with WaterMark sensor attached](image1)

*Figure 3. SmartFarm board v5.4 with WaterMark sensor attached*

4. The image below summarizes what the SmartFarm board should like with each of the three sensors configured:

![SmartFarm board v5.4 with all three types of sensors attached](image2)

*Figure 4. SmartFarm board v5.4 with all three types of sensors attached*

5. Ensure that a micro-SD card is inserted into the SD card slot on board. This slot is located just above the data pins.
6. Ensure that the XBee receiver radio is plugged into the associated pins, as shown by any of the images above.
7. Ensure that there is either a battery or voltage source plugged into the board.
8. Plug the XBee Wireless Programming Coordinator into an open USB slot on the PC with the SmartFarmIDE installed.

**SmartFarmIDE Configuration:**

1. Follow the steps for “Board Configuration” before completing the steps listed here
2. Navigate on the PC to where the SmartFarmIDE folder is installed.
3. Within the SmartFarmIDE folder
a. Navigate to Examples->MeasurementExamples->takeMeasurements
b. In this folder, there should be a takeMeasurements.ino file
c. Right click on this file and select Open With->SmartFarmIDE
d. The SmartFarmIDE should open with the default sketch
e. The image below shows what the sketch should look like:

![Figure 5. Ideal sketch](image)

4. Power on the SmartFarm Board
5. Press the “Upload” button in the SmartFarmIDE to upload the sketch to the board. The upload button is located in the top left corner of the IDE. This button is denoted as: 🔄
6. Once the sketch has been uploaded to the board, open the Serial Monitor by pressing the “Serial Monitor” button in the IDE. This button is located in the top right corner of the IDE. This button is denoted as: 🔄
7. Once the Serial Monitor is open, when the SmartFarm board goes into a measurement phase, data should be printed in the general format below:

```
Finding temperature sensors: Found 0 devices.
Scanning Decagon 5TE addresses.
No DECAGON 5TE sensors found.
Found 0 Decagon 5TE sensors.
Reading Watermark sensors...
NA NA 135555.160 NA
Reading temperature sensors...
Reading Decagon 5TE Sensors...
Format: (Time elapsed), (Sensor ID, Type, and Version). (Measurements)
Upload
```

![Figure 6. Serial Printing of Measurement Data](image)

b. **SmartFarm Base Station Measurement and Uploading**

Refer to Raspberry Pi Smart Farm Base Station for details on how to setup and configure the Base Station before performing these steps. Also see appendices on page 39 for details on how the scripts
function. The Base Station will automatically post data on startup. After uploading a program, it will return to posting data.

**Required Materials**

1. SmartFarm v5.4 board with takeMeasurements.ino* (or similar. See Wirelessly Programming a SmartFarm Board on the Next Wake Cycle)
2. 3.7v LiPO Battery
3. Decagon 5TE EC sensor
4. DallasTemperature DS18B20 Temperature Sensor
5. WaterMark soil moisture sensor
6. XBee Wireless Programming Pair
7. Raspberry Pi Model B**
8. Cat5e Ethernet cable
9. 5V Micro USB Power Supply

*The upload will occur on the next wake cycle if the board has the takeMeasurements.ino file programmed onto it, as this program includes the trigger data.

** Testing has only occurred with the Model B. Though this setup walks through with the Model B, but may also work with other Raspberry Pi models.

**Uploading on Next Wake Cycle**

1. Connect to the SmartFarm Base Station Website (Type the IP address of the Base Station into the browser)
2. Unlock the password protected upload page
3. Select an ino file to upload to the SmartFarm board (takeMeasurements.ino is a good choice)
4. Choose the appropriate directory (the one in which is being watched by the python script)
5. Click Upload
   a. The board will flash rapidly just as it does with the IDE

**Posting Data on Cloud data host ThingSpeak**

After uploading script, the Base Station will return to data collection and posting to ThingSpeak without user intervention.

**III. Plug and Play Library for the SmartFarmIDE**

a. **Overview**

The overall expectation of this product is to create an autonomous Arduino library that accurately and robustly collects plant environment data from the Smart Farm board. The Smart Farm PCB is split into two primary functions: Data Acquisition(DAQ) and Power Control. Each function of the PCB is controlled by an Atmega 328P Microcontroller. The primary concern for this report is the functionality of the DAQ microcontroller.

The DAQ microcontrollers’ current role in the Smart Farm project is to regularly take data readings from three different types of sensors. The three types of sensors are the Decagon 5TE
Electrical Conductivity Sensor, the Waterproof Dallas Temperature Sensor (DS18B20), and lastly the Watermark Soil Moisture Sensor (200SS). The specific functionality of each of these sensors is explained in further detail later in this report.

The desired functionality of the SmartFarmMeasure library is to be plug and play. The library also contains functions for ease of use, upon user desire of creating their own sketches for interfacing with the SmartFarm board.

**b. Approach**

The designated approach for accomplishing the goal of this project is iterative, or modular. For the sake of clarification, the project has been broken down into the following steps over the course of twenty weeks, and are to be implemented in order.

1. Watermark Sensor Integration
   a. Create Arduino Sketch and take successful readings from Watermark sensor through an Arduino Genuino board
   b. Transfer Arduino sketch code into portable library equivalent function

2. Waterproof Dallas Temperature Sensor Integration
   a. Create Arduino Sketch and take successful readings from Dallas Temperature sensor through Arduino Genuino board
   b. Transfer Arduino sketch code into portable library equivalent function

3. Decagon 5TE Sensor Integration
   a. Create Arduino Sketch and take successful readings from Decagon 5TE sensor through Arduino Genuino board
   b. Transfer Arduino sketch code into portable library equivalent function

4. Waterproof Dallas Temperature Sensor Single-pin Integration
   a. Create an Arduino Sketch that utilizes both the Dallas Temperature Arduino Library and the OneWire Arduino library to successfully read from multiple temperature sensors sharing a single data bus
   b. Transfer Arduino sketch code into portable library equivalent function

5. Decagon 5TE Sensor Single-pin Integration
   a. Create an Arduino Sketch that utilizes the Arduino SDI-12 library to successfully read from multiple Decagon sensors sharing a single data bus
   b. Transfer Arduino Sketch code into portable library equivalent function

6. Implementation of wrapper functions for clarity and ease of use.

**c. Hardware**

**Decagon 5TE Electrical Conductivity Sensor**

Data Sheet:


Image:
Dallas Temperature Waterproof Temperature Sensor

Data sheet:


Image:

Watermark Soil Moisture Sensor

Data sheet:

http://www.specmeters.com/assets/1/22/6450WD.pdf

Image:
d. **Software**

The following section will provide a breakdown of the functions in the SmartFarmMeasure library. The functions within the library can be broken up into three tiers: High Level User Functions, Mid Level User Functions, and Low Level Functions. Each tier will be explained in terms of functionality to the user.

**High Level User Functions**

The following list contains each of the high level user functions and their descriptions:

1. `SmartFarmMeasure SmartFarmMeasure();`
   a. This is the constructor for the SmartFarmMeasure object.
   b. Any time an instance of the SmartFarmMeasure object is declared, this method is called.
   c. This method contains no functional logic, but returns a SmartFarmMeasure object to the user.

2. `void setupAll();`
   a. This method is used to setup each of the three different types of sensors supported by SmartFarm.
   b. This function will sequentially initialize the Temperature sensors, Decagon sensors, and the WaterMark sensors.
   c. This function will send a signal to the MCU to turn off wireless programming.
   d. This method does not return anything, and should be called in the “setup()” function of a typical Arduino sketch.
   e. This method will print initialization data both to the Serial Monitor and the SD card, if there is one.

3. `void runAll();`
   a. This method is used to take readings from each of the three different types of sensors supported by SmartFarm.
   b. This method should only be called after setupAll has had a chance to initialize the sensors.
   c. This method does not return anything, and should be called within the “loop()” function of a typical Arduino sketch.
   d. This method will print data read from each of the sensors both to the Serial Monitor and the SD card, if there is one.

**Mid Level User Functions**

The following list contains each of the mid level user functions and their descriptions:

1. `void setupWM();`
   a. This method essential for properly running the SmartFarm board. It sends a series of signals to the MCU that will turn off wireless programming.
   b. This method functionally prepares the necessary digital pins for the voltage divider used to read the WaterMark sensors.
c. This method must be called before using “readWM()”.
d. This method does not return anything, but rather outputs results where necessary.

2. void readWM();
   a. This method is used for taking measurements from each of the analog pins on the SmartFarm board.
   b. In the case that there is no WaterMark sensor attached, “NA” is outputted.
   c. This function writes read data both to the Serial Monitor and the SD card, if there is one.
   d. This function utilizes an on board voltage divider to take accurate measurements of the variable resistor (WaterMark sensor).
   e. This method does not return anything, but rather outputs results where necessary.

3. void setupTemps();
   a. This method is used for initializing the DS18B20 temperature sensors.
   b. This method will automatically search the shared data bus of the temperature sensors and allow it’s subsequent functions to take readings from each.
   c. This method will output initialization data to both the Serial Monitor and the SD card, if there is one.
   d. This method must be called before the readTemps() function can be used.

4. void readTemps();
   a. This method is used for reading temperature data from the initialized DS18B20 temperature sensors.
   b. This method will take a reading from each of the temperature sensors found during initialization.
   c. This method will output data both to the Serial Monitor and the SD card, if there is one.
   d. The setupTemps() method must be called before this function can be used.

5. void setupDecSensors();
   a. This method is used to initialize the Decagon 5TE EC sensors.
   b. This method will automatically search the shared data bus for the Decagon sensors and allow subsequent functions to take readings from each sensor found.
   c. This method will output initialization data both to the Serial Monitor and to the SD card, if there is one.
   d. This method must be called before the readDecSensors() method can be used.

6. void readDecSensors();
   a. This method is used to take Decagon readings from each of the Decagon sensors found during initialization.
   b. This method will output data both to the Serial Monitor and the SD card, if there is one.
c. The setupDecSensors() method must be called before this function can be used.

7. void setupSD();
   a. This method is used to initialize the SD card on the SmartFarm board.
   b. This method must be called before and writing or reading from the SD card can occur.

8. void write2SD(String dataString);
   a. This method is used to generically write a String of data to the SD card.
   b. The setupSD() method must be called before this function can be used.
   c. If there is no SD card found, a relevant message will be outputted to the Serial Monitor.

**Low Level Functions**

The following list contains each of the low level functions and their descriptions. Note: these are private methods, and cannot be called from outside the library. This is the definition of the Low Level tier of functions.

1. String getDevAddress(DeviceAddress deviceAddress);
   a. This function is used by the Temperature sensors.
   b. This function takes a DeviceAddress object and returns a String representation of it.

2. void printData(DeviceAddress deviceAddress);
   a. This function will read and print data from a specific temperature sensor.
   b. This function will write data to the Serial Monitor and the SD card, if there is one.

3. void takeDecMeasurement(char i);
   a. This function will read data from a specific Decagon sensor.
   b. The parameter “i” represents the unique byte address of the sensor desired to be read.
   c. This function will output read data both to the Serial Monitor and the SD card, if there is one.

4. void printBufferToScreen();
   a. This method is used by the measurement methods of the Decagon sensors.
   b. This method will print the data read from a Decagon sensor both to the Serial Monitor and the SD card, if there is one.

5. boolean checkActive(char i);
   a. This method will check if a Decagon device associated with the unique byte address “i” is currently interfaced with the SmartFarm board.
   b. This method returns a Boolean value, true if there is a Decagon sensor with the associated address, and false if otherwise.

6. boolean setTaken(byte i);
   a. This method will set the Decagon byte address represented by “i” to active in the Decagon register.
b. This method returns a Boolean value, true if the address is available and set, and false if the address is already taken.

7. boolean setVacant(byte i);
   a. This method will set the Decagon byte address represented by “i” to vacant in the Decagon register.
   b. This method returns a Boolean value, true if the address was taken and set to vacant, false if the address was already vacant.

8. boolean isTaken(byte i);
   a. This method checks if the Decagon byte address represented by “i” us currently listed in the Decagon register.
   b. This method returns a Boolean value, true if the address is taken and false if otherwise.

9. char printInfo(char i);
   a. This method will print information associated with the Decagon device with the byte address represented by “i”.
   b. The information printed consists of the Decagon family, the devices firmware revision, and the unique byte address.
   c. This method will print to both the Serial Monitor and the SD card, if there is one.

10. byte charToDec(char i);
    a. This method is used for converting the character address “i” for a Decagon sensor to a raw value. The raw value is returned.

11. char decToChar(byte i);
    a. This method is used for converting the raw address “i” for a Decagon sensor to a char value. The char value is returned.

12. void delayMilliseconds(int x);
    a. This method is used for delaying x number of milliseconds.

13. void finishUp();
    a. This method is used for the sole purpose of stopping wireless programming from the MCU.

IV. **Raspberry Pi Smart Farm Base Station**

This walkthrough provides General configuration of Raspberry Pi, General configuration of SmartFarm Base Station, Wirelessly Programming a SmartFarm Board on the Next Wake Cycle, and Collecting SmartFarm Data and Posting to the Web.

*NOTE: A windows based pc was used in conjunction with the Raspberry Pi for setup*
a. **General configuration of Raspberry Pi Model B**

To configure the Raspberry Pi for the first time, it is best to follow the guides from the Raspberry Pi Foundation website as these are very well documented and easy to read. The Raspberry Pi is a very small form factor PC which can run a host of operating systems. The NOOBS is used in this walkthrough and is very easy to get started quickly.

For this walkthrough, the Raspberry Pi will first be setup as a desktop pc, with keyboard, mouse, and monitor. It will then be configured to run “headless” through Secure shell (SSH) to a laptop via only an Ethernet cable and micro USB cable for power, which eliminates the need for the keyboard, mouse, and monitor. See Running the Raspberry Pi headless (without monitor, keyboard, or mouse).

**Software and downloading the OS**

The Raspberry Pi foundation has made a great startup guide starting with the software. Follow this guide to install the OS.


**Guide to setting up the pi for the first time**

After the downloading the operating system onto the SD card, the peripherals need to be connected to run the Raspberry Pi as a desktop computer.

Check if you have the right equipment.


Continue onto making the connections.


**Connecting the Pi to the internet**

After initial setup of the Raspberry Pi, it needs to be connected to the internet to receive the latest updates. Connecting the Raspberry Pi to the internet


Follow this guide to update the Pi system.


```
pi@SmartFarmBaseStation:~ $ sudo apt-get update
```

*Figure 7. Updating the Raspberry Pi: sudo apt-get update*

```
pi@SmartFarmBaseStation:~ $ sudo apt-get upgrade
```

*Figure 8. Upgrading the Raspberry Pi: sudo apt-get upgrade*

The Pi has been setup and is now ready to be utilized. With its current setup, however, the Pi needs a mouse, keyboard and a monitor. For the SmartFarm Base Station, the Pi needs to be setup without the external devices or running “headless”.
**Running the Raspberry Pi headless (without monitor, keyboard, or mouse)**

Running the Raspberry Pi headless is to make better use of it for the SmartFarm. A headless Raspberry Pi is without monitor, keyboard or mouse and can be used with a laptop or desktop pc. The way to do this is via Secure Shell (SSH). A tutorial is made by Adafruit and can be found at

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-6-using-ssh

Once the SSH is enabled on the Raspberry Pi, then the keyboard, mouse, and monitor can be disconnected.

Headless setup uses only an Ethernet cable and microUSB cable for power (Figure 9).

---

**Figure 9. Headless Raspberry Pi**

By following this guide, you will need to download PuTTY on the host computer, found at

http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

---

**Figure 10. The PuTTY icon**

To use PuTTY, the IP address of the Raspberry Pi is needed. If the IP address of the Raspberry Pi cannot easily be found, then the hostname can be used instead (Figure 11). This only works if either iTunes or Bonjour Service is installed on the host computer. Download and install either on the host computer to use the hostname.

Bonjour Service

iTunes
By having either of these installed on the host computer, the hostname of the raspberry is easily found. Now connect to the Raspberry Pi via PuTTY.

**Figure 11. Using the Raspberry Pi Hostname to connect via PuTTY**

The standard username and password for the Raspberry Pi is `pi` and `raspberry` respectively. Once the Raspberry Pi is setup, there are additional steps to use it as a base station for SmartFarm.

**Giving the Raspberry Pi a Static IP Address**

Further setup of the Raspberry Pi includes setting a static IP address so that it can be connected to the home router internet connection and can provide remote access. A quick google search will help with this, but the essential steps are listed here. The dhcpcd.conf file needs to be edited and can be opened in the terminal by typing in the command shown in Figure 12. Editing the dhcpcd.conf file for Static IP

```
pi@SmartFarmBaseStation:~ $ sudo nano dhcpcd.conf
```

**Figure 12. Editing the dhcpcd.conf file for Static IP**

At the bottom of the file view these lines and change them to your desired address. The eth0 is for a wired Ethernet connection to the router and the wlan0 is for a wifi router connection. To set a static IP address, set: `static ip_address =` to your desired static IP address. leave the 24 at the end of it. The static routers is set to the address of your gateway which is likely your router. The static domain_name_servers = is the IP address of your DNS, set it to your router.
Figure 13. dhcpd.conf in nano editor with ethernet eth0 and wifi wlan0

After the three addresses of either wlan0 or eth0 have been set, save the file and close the editor with Ctrl-X, Y, Enter. Perform a reboot on the Raspberry Pi and check that the inet addr: has changed with

```
pi@SmartFarmBaseStation:~ $ ifconfig
```

Figure 14. Command to check for the static IP address | ifconfig

Now that a static IP address is setup on the Raspberry Pi, the Base Station can be connected from anywhere on your network instead of just on your computer. Connecting to the server through the web browser is shown in Uploading a File from the WordPress Site to the Base Station.

b. **General configuration of SmartFarm Base Station**

To use the Raspberry Pi as a Base Station for SmartFarm, a set of additional configuration steps need to be taken. The SmartFarm Base Station needs to be configured as a web server, to host a website to develop the WebIDE. The web server used here is Wordpres, an open source software to create a website.

Along with setting up the web server, the Base Station also needs to be able to upload programs to the SmartFarm board and collect measurement data. PlatformIO is a Professional development environment that will be used to compile and upload programs to the SmartFarm v5.4 board. The environment is supported by many embedded devices manufacturers and is also open source. For more info refer to http://platformio.org/get-started. For the SmartFarm Base Station, PlatformIO Core was downloaded, installed, and setup to upload programs to the board. To collect measurement data a hosting site, ThingSpeak.com is utilized. See Collecting SmartFarm Data and Posting to the Web.

Before setting up the web server and platformIO on the base station, it is worthy to take a moment and mention two programs that will make working with the Base Station a little nicer. Transferring files from your computer to your Raspberry Pi Base Station can be a little bit of a task. To transfer files from the computer to the Raspberry Pi, a program called Windows Secure Copy was used, which allows the user to view and manipulate the file system on the Raspberry Pi. By writing programs and then saving them in WinSCP, this also saves them to the Raspberry Pi and allows making changes to programs easy.
The hidden Raspberry Pi files and directories should also be shown in WinSCP. To unhide these files go to Options > Preferences > Panels and check the Show hidden files box. Figure 17 and Figure 18.
Another program worth giving mention to is Notepad++, which gives a clean environment to write programs in the language of your choice. By using both WinSCP and Notepad++ together, editing and troubleshooting programs becomes streamlined.

Figure 18. Showing hidden files in WinSCP: Panels, Check Show hidden files box and Click OK

Figure 19. The Notepad++ icon and download at https://notepad-plus-plus.org/download/v7.3.3.html

Figure 20. Screenshot of using Notepad++ to edit python script files.
PlatformIO Core 3.4 Installation and Uploading

From the PlatformIO website download the “get-platformio.py” script. Look for the link.

![Local Download (Mac / Linux / Windows)](image)

To install or upgrade PlatformIO, download (save as...) get-platformio.py script.

**Figure 21. get-platformio.py script download**

Use WinSCP to transfer the file from the host computer to the Raspberry Pi into the desired directory. /home/pi is used here.

![Copy the get-platformio.py from your PC to the Raspberry Pi](image)

**Figure 22. Transfer file from computer to Raspberry Pi using WinSCP**

Run the get-platformio.py script to install PlatformIO Core 3.4

```
pi@SmartFarmBaseStation:~ $ python get-platformio.py
```

**Figure 23. Run the Command to install PlatformIO Core**

Next, the SmartFarmIDE files need to be added into the PlatformIO directory. Using WinSCP copy the SmartFarmIDE files into the respective directories. Look for the .platformio directory on the Raspberry Pi. It should be located under /home/pi. Navigate to the directory /home/pi/.platformio/packages/framework-arduinoavr. It should look something like Figure 24.
There are three changes in this directory: smartfarm cores, smartfarm variants, and the boards.txt. They need to be added into the respective directories.

Figure 25. Copying the variants, cores, and boards.txt into the /home/pi/.platformio/packages/framework-arduinoavr directory
To check if these changes were successful, see if the platformIO recognizes the smartfarm board by running the command in Figure 26. The smartfarm board will be shown in the list if the files were copied over properly, Figure 27

```
pi@SmartFarmBaseStation:~/Another_PROJECT $ platformio boards smartfarm
```

*Figure 26. PlatformIO boards recognition command*

```
pi@SmartFarmBaseStation:~/Another_PROJECT $ platformio boards smartfarm
```

<table>
<thead>
<tr>
<th>ID</th>
<th>MCU</th>
<th>Frequency</th>
<th>Flash</th>
<th>RAM</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATMEGA328P</td>
<td>8MHz</td>
<td>3kB</td>
<td>2kB</td>
<td>SmartFarm Board v5.4</td>
</tr>
</tbody>
</table>

*Figure 27. The SmartFarm board is recognized and ready to be initialized*

The PlatformIO project can now be initialized for SmartFarm board. First make the desired directory of the project by calling the mkdir command. Any directory name can be given, SmartFarm_CurrentDAQ is used here as it indicates the currently loaded DAQ sketch is in this project directory, Figure 28. After creating the directory, change into it as your working directory, Figure 29. Now initialize the PlatformIO project by running the command in Figure 30.

```
pi@SmartFarmBaseStation:~/Another_PROJECT $ mkdir SmartFarm_CurrentDAQ
```

*Figure 28. Create the directory for a new PlatformIO project*

```
pi@SmartFarmBaseStation:~/SmartFarm_CurrentDAQ $ cd SmartFarm_CurrentDAQ/
```

*Figure 29. Change into the PlatformIO project directory just created*

While in this directory run the command in to initialize the platformio project.

```
pi@SmartFarmBaseStation:~/SmartFarm_CurrentDAQ $ platformio init --board smartfarm
```

*Figure 30. Initialize the PlatformIO project for smartfarm board*

The project directory will now contain three directories (one is hidden) and platformio.ini, which is the initialization configuration file. The src directory and lib directory are the important ones for smartfarm. The src directory will contain the sketch to upload to the SmartFarm board while the lib directory will contain the new SmartFarm libraries discussed in Plug and Play Library for the SmartFarmIDE on page 10. Copy and paste the libraries from the SmartFarmIDE into the lib directory. If done properly, running the command in the lib will list the libraries for smartfarm, Figure 31.

```
pi@SmartFarmBaseStation:~/SmartFarm_CurrentDAQ/lib $ ls
DallasTemperature DSS231 OneWire readme.txt RTClib SD SD112 SmartFarmMeasure SPI Wire
```

*Figure 31. Run ls command in the lib directory to confirm the libraries needed to upload to SmartFarm*

Almost time to upload. The src directory needs a sketch to upload to the SmartFarm board. Copy and paste the example sketch takeMeasurements.ino from the SmartFarmIDE into the src directory.
Upload Test to the SmartFarm board from the Raspberry Pi Base Station

Now to test uploading to the SmartFarm board with platformIO and the Raspberry Pi. Connect the USB XBee Dongle to an empty port on the Raspberry Pi. Connect the coordinator radio to the dongle. make sure to follow the pattern outlined on the dongle for which direction to place the XBee. Setup the SmartFarm board with the remote radio and battery (the sensors are not needed yet as this is a test to make sure platformIO works correctly to upload).

First compile the takeMeasurements.ino file in the src directory by running the command in Figure 32:

```
pi@SmartFarmBaseStation:~/SmartFarm_CurrentDAQ/lib $ platformio run --e smartfarm
```

*Figure 32. Compile the sketch in the src directory of the PlatformIO project*

Then once it has successfully compiled the code, upload it to the SmartFarm board by running the command in Figure 33:

```
pi@SmartFarmBaseStation:~/SmartFarm_CurrentDAQ/lib $ platformio run --e smartfarm --t upload
```

*Figure 33. Upload the sketch in the src directory of the PlatformIO project to the SmartFarm v5.4 board*

The timing on upload is still a bit of a trick, as the upload has to occur just when measurement mode is finished on the smartfarm board. If the programming window was missed, push the reset button on the SmartFarm board and carefully try again. The SmartFarm board v5.4 blinks rapidly when being programmed, blink normally after, and then return to sleeping status.

A simple python script, Figure 35, can let you know if the programming worked or not, which opens a serial port and receives the serial data from the SmartFarm board. The serial data is seen in the command line when the script is running, Figure 34. To stop the python script in the terminal, press Ctrl-C; this will return you to the command line.

If the serial is not working as expected, or the script returns errors, then the Raspberry Pi might need the serial library. The library used here is python-serial and is imported at the top of the script. Run the installation command, Figure 36.
Figure 34. Raspberry Pi is receiving serial data from the SmartFarm board with no sensors connected

```python
import serial
import os
serialPort = '/dev/ttyUSB0'
serialBaudRate = 57600
ArduinoSerialData = serial.Serial(serialPort, serialBaudRate)
ArduinoSerialData.flushInput()
ArduinoSerialData.flushOutput()
try:
    while True:
        words = ArduinoSerialData.readline()
        print words
except KeyboardInterrupt:
    os._exit(-1)
```

Figure 35. Simple serial data python script Arduino_serial.py

Figure 36. Install python-serial: sudo apt-get install python-serial

Configuration of SmartFarm Base Station as a WordPress Web Server

The Raspberry Pi Base Station setup as a web server gives the capabilities of remote access and therefore the foundation of a web IDE. The Raspberry Pi organization has developed a guide to build a web server with Apache, MySQL, and PHP. The guide is found at https://www.raspberrypi.org/learning/lamp-web-server-with-wordpress/worksheet/

This guide is a step by step walkthrough of setting up the web server. It will be briefed here. Install the apache2 package by running the command in Figure 37.
Test the web server by opening a web browser in another computer, or this host computer if it is on the same network as the Raspberry Pi, and typing in your Raspberry Pi’s IP address. The address can be found by typing in the command in Error! Reference source not found.. The Apache2 Debian Default Page should come up if the connections are setup right. If not, make sure to go through Giving the Raspberry Pi a Static IP Address.

Wordpress is written in PHP, so it needs to be installed on the Raspberry Pi.

Change the working directory to the html directory, Figure 41, and create an index.php file with the command in Figure 42 The index.php could contain something like <?php echo “hello world”; ?>. Save the file by pushing Ctrl-X, Y, Enter. Remove the index.html file as it takes priority over php, Figure 43.

The index.php file will now print hello world to the browser after refresh. If it doesn’t, restart the pache2 service as in Figure 44.

Next is to install MySQL and php-MySQL packages. When installing, MySQL asks for a password to give website access to the database. Keep this password handy.
Figure 45. Installing MySQL and php-mysql packages

Restart apache service, Figure 44.

Remain in the /var/www/html directory and remove all the files in this directory. Download WordPress by following the command in Install WordPress and cleanup the downloaded folder,

```bash
pi@SmartFarmBaseStation:/var/www/html $ sudo rm *
```

Figure 46. Remove all contents in the /var/www/html directory before downloading WordPress

```bash
pi@SmartFarmBaseStation:/var/www/html $ sudo wget http://wordpress.org/latest.tar.gz
```

Figure 47. Download WordPress

```bash
pi@SmartFarmBaseStation:/var/www/html $ sudo tar xzf latest.tar.gz
```

Figure 48. Extract the downloaded tarball

```bash
pi@SmartFarmBaseStation:/var/www/html $ sudo mv wordpress/*
```

Figure 49. Move the folder to the current working directory

```bash
pi@SmartFarmBaseStation:/var/www/html $ sudo rm -rf wordpress latest.tar.gz
```

Figure 50. Remove the now empty folder to cleanup

Now that Wordpress is installed, the ownership of it should be handed off to the Apache user,

```bash
pi@SmartFarmBaseStation:/var/www/html $ sudo chown -R www-data:.
```

Figure 51. Change ownership of the wordpress files to the Apache user

Now to get the WordPress site setup. A database from the mysql package is needed, start one by typing in the command in … This will request a password, which is the same one setup in 39

```bash
pi@SmartFarmBaseStation:/var/www/html $ mysql -uroot -p
```

Figure 52. Start a mysql database

Create the database with the command in … This command should give a success message of “Query OK, 1 row affected (0.00 sec)” Now exit the MySQL with Ctrl-D

```bash
pi@SmartFarmBaseStation:/var/www/html $ mysql> create database wordpress;
```

Figure 53. Creating the mysql database for Wordpress site

WordPress configuration and website setup takes place on the browser. If all went well setting up the Apache server, WordPress site and MySQL, the browser session of your Raspberry Pi’s IP address should show a welcome to WordPress page. At the bottom of this page, click the Let’s Go! button
WordPress will ask for some names and credentials. Fill them out as shown in … Click submit and run install. Give a site a title, and create an account with username, password, and email address. Click Install WordPress.

Table 1. Naming and credentialing the database for WordPress

<table>
<thead>
<tr>
<th>Database Name:</th>
<th>wordpress</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name:</td>
<td>root</td>
</tr>
<tr>
<td>Password:</td>
<td>&lt;your password here&gt;</td>
</tr>
<tr>
<td>Database Host:</td>
<td>localhost</td>
</tr>
<tr>
<td>Table Prefix:</td>
<td>wp_</td>
</tr>
</tbody>
</table>

The WordPress site is now viewable and “online” to your private network. To login in again, type http://Your_IP_address/wp-admin from another browser or computer.

Permalinks make the URL’s user friendly and is recommended to set them. Go to your WordPress site, login and go to dashboard.
Now to enable the Apache’s rewrite mod,

```
pi@SmartFarmBaseStation:/var/www/html $ sudo a2enmod rewrite
```

Figure 57. Apache’s rewrite mod enabled command
Also, edit the Apache configuration file, … Edit the lines after line 1 <VirtulHost *:80> to those shown in …

```
pi@SmartFarmBaseStation:/var/www/html & sudo nano /etc/apache2/sites-available/000-default.conf
```

*Figure 58. Edit the Apache configuration file in the nano editor*

![Example Apache configuration line]

*Figure 59. Lines edited in Apache configuration file*

After closing and saving, restart Apache for changes to take effect. …

**Uploading a File from the WordPress Site to the Base Station**

The WordPress site is now ready to be customized. This means pictures, functions, information, controls, and the like can be done with the WordPress site. Lots of documentation is available on the web, and since it is open source, it is freely available and adaptable.

The main function for the SmartFarm Base Station’s website is to upload a program from anywhere to the SmartFarm board and provide a visual for the data. To do this, a plugin is used called WordPress File Upload.

```
Figure 60. Install the WordPress File Upload plugin from the Add New under Plugins
```

To use the uploads, navigate to /var/www/html/wp-content and notice the uploads when the ls command is used to list the contents of wp-content. The files from the wordpress site will go into this directory, unless another directory is specified. To add another directory here, use the command mkdir directory_name. Once created, the ownership of it must be changed just like the in 45... An example is sudo chown -R www-data: . This gives access to the directory from the server, to be able to post to it from the website,
To make the file uploader work with .ino files or SmartFarm programs, the blacklist needs to be edited. Follow the command in 56 to

```
pi@SmartFarmBaseStation:/var/www/html/wp-content/plugins/wp-file-upload/lib
```

Edit this wfu_security.php file with the command .. The $wfu_extension_blacklist = array("2clk" => 1, "386" => 1, "3dfbat"
… ) line needs to be edited to allow .ino files. Scroll to the right to find the “.ino” and delete it from the list. Type Ctrl-W to locate ino and then delete the entry.

```
pi@SmartFarmBaseStation:/var/www/html/wp-content/uploads $ sudo chown -R www-data: .
```

**Figure 61. Change the ownership of the directory to include the new directory**

**Figure 62. Navigate to the lib folder of the WordPress File Upload plugin**

**Figure 63. List the contents of lib and find the wfu_security.php file**

**Figure 64. Locating the ino in the blacklist press Ctrl-W**
Figure 65. Delete the entry once found, from the first comma to the 1. Ex: ..., "inl" => 1, "ino" => 1, "inp" => 1, goes to ..., "inl" => 1, "inp" => 1, ...

Click on customize your site, and select a place to edit, add a new page, title it appropriately, click Save and publish, and close out of the theme customization.

At the Dashboard, to add the File Upload plugin instance, go to Settings > Wordpress File Upload. Go to the bottom and Add Plugin Instance, change the shortcode to reflect the options for the plugin,
Figure 67. Add the Plugin Instance to the newly created Page

The File Upload is now ready to be used on the SmartFarmBaseStation site. For security purposes, however, uploading on WordPress should require a password. More security precautions can be taken, but a minimum is having a password to access the upload button. To add this feature to the page, go to the WordPress Dashboard, click Pages > All Pages then edit the Upload page. On the right-hand side, under Publish is a Visibility setting that allows password protection. The Upload page can also be edited by changing the shortcode here. The shortcode is a place where the configuration for how the Upload functions.

Figure 68. Editing to add a password protected Upload and Configuration
The Upload page is now complete and ready to be utilized, however, it is not yet ready to upload to a SmartFarm board. So far the Upload page on the website only gets a file onto the Raspberry Pi. The SmartFarmBaseStation needs a python program to take the uploaded file from the web and then upload it to the SmartFarm board. See...

![Image of Upload page](image)

Figure 69. Upload File on the WordPress site

c. **Wirelessly Programming a SmartFarm Board on the Next Wake Cycle**

To upload a new program from the Raspberry Pi base station to a SmartFarm v5.4 node on the next wake cycle, the board uses a trigger command from the SmartFarm board. This trigger is the serial data from the board, which is recognized in the python script. The board sends the upload trigger when it wakes up as part of the measurement data cycle. A python program on the base station receives this trigger command from the node and uses the PlatformIO project to upload the new program to the board. For the board to send this trigger, it must first be programmed with the takeMeasurements.ino or a similar program that includes the trigger, see **Software**. Once the ino file is uploaded to the node, a call of the UPLOAD_CurrentDAQ.py from the base station will upload a new program to a SmartFarm node on the next wake cycle from the WordPress site, see UPLOAD_CurrentDAQ.py calls compile and upload using PlatformIO.

d. **Collecting SmartFarm Data and Posting to the Web**

Collecting data from the SmartFarm v5.4 nodes and posting it to the web requires a host. ThingSpeak is easily utilized to post data to the web (Figure 70). ThingSpeak is a trademark of MathWorks, which are the makers of MATLAB and Simulink. Future analysis of the data posted in ThingSpeak can easily utilize
MATLAB. The free service of ThingSpeak allows ~8,200 messages/day and <3 million/year, posting data every 15 seconds. Create an account at https://thingspeak.com/.

![ThingSpeak Trademark by MathWorks](image)

*Figure 70. The ThingSpeak Trademark by MathWorks*

When an account is created on ThingSpeak, a channel must be created as well. The channel takes up to 8 fields of data, or more so 8 inputs. Meaning that only 8 measurements can be posted at a time. For example, 1 channel can take 4 Watermark sensors and 4 temperature sensors. Another channel would then have to be created if more sensor readings are desired. It might be that a farmer would give each SmartFarm board its own channel. Name the channel, describe it and enter a Field Label for each Field. The Fields will be the individual sensors on the SmartFarm board.

To write data to the ThingSpeak channel, an API write key is provided for each channel. A python program, ThingSpeak.py, is used to post the data received from a SmartFarm node to ThingSpeak via the channels API write key. See ThingSpeakCode.py posts data to ThingSpeak.com.

**Sample SmartFarm Data Posted to ThingSpeak**

The Watermark sensor reads 0-100 centibar (kPa), 0 centibar being wet, and 100 centibar being dry. To emulate this, the sample data was collected overnight with a fan blowing over the node, which read the Watermark sensor ports A0 – A4.

![Sample SmartFarm WaterMark Data posted on ThingSpeak](image)

*Figure 71. Sample SmartFarm WaterMark Data posted on ThingSpeak*
V. **Overall Improvements**

1. Changing to another node. Implementing X-CTU on the Pi. This may need access to host radio to change which node/radio to program. Maybe from the XBee radio build table.

2. Configuring the radios via Raspberry Pi Automatically so that they work and that the End user doesn’t have to mess with configurations. Plug and Play scenario: user plugs in the external radio, the Pi configures it for the network, the user then plugs it onto the SmartFarm board. User confirms data with a test

3. In UPLOAD_CurrentDAQ.py, need an error checker to determine upload success or fail, which can then be directed to the user via WordPress, as right now the user has no feedback of what happens when the program is uploaded.

4. Security

5. WebIDE Development

6. Virtual Private Network for secure remote access
/*
 * This header file contains all function definitions
 * for the SmartFarmMeasure Data Acquisition library.
 * The functions are logically separated into public
 * and private notation, representing what is seen
 * and used user side, versus what is used by the
 * library internally. For further details on each
 * of the functions declared in this file see
 * SmartFarmMeasure.cpp source file.
 *
 * Author: Nathan Oto
 * Project: SmartFarm
 * Professor: Bo Liu
 * Date: 6/11/2017
 */

#ifndef SmartFarmMeasure_H
#define SmartFarmMeasure_H

//The OneWire library is used by DallasTemperature
//and needed for interfacing with the temperature
//sensors utilized for this project.
#include <OneWire.h>

//The DallasTemperature library is used to interface
//with the temperature sensors on a single dig pin.
#include <DallasTemperature.h>

//The SD library is used for writing data read from
//each of the sensors to a micro-SD card on-board.
#include <SD.h>

//SPI is the communication protocol utilized throughout
//the SmartFarm project.
#include <SPI.h>

//Wire is a basic library used by SmartFarm in a variety
//of subsequent functions.
#include <Wire.h>

//The SDI12 library is needed specifically for reading from
//the Decagon Electrical Conductivity sensors all connected
//to a single digital pin.
#include <SDI12.h>

//The core library for Arduino.
#include <Arduino.h>

//The declaration of the SmartFarmMeasure Class
//which must be used to utilize it's subsequent functions.
//Example: "SmartFarmMeasure mysmf;"
class SmartFarmMeasure {
    //The public function declarations that can be
    //publicly seen and used by the user.
    public:
        //The constructor for the SmartFarmMeasure class.
        SmartFarmMeasure();
        //Wrapper function for initializing the temperature
        //sensors on a OneWire bus.
        void setupTemps();
        //Wrapper function for reading temperature sensors
        //already initialized on the OneWire bus.
        void readTemps();
        //Wrapper function for initializing the SD card
        //interfaced to the SmartFarm board.
        void setupSD();
        //Function to write to the initialized SD card.
        void write2SD(String dataString);
        //Wrapper function to initialize the Decagon Sensors
        //on a single digital pin using the SDI12 library.
        void setupDecSensors();
        //Wrapper function for reading from the initialized
        //Decagon Sensors interfaced to a single digital pin.
        void readDecSensors();
        //Wrapper function to initialize the WaterMark sensors
        //interfaced to the analog pins.
        void setupWM();
        //Wrapper function to read from the WaterMark sensors
        //interface to the analog pins
        void readWM();
        //Wrapper function for initializing each of the three
        //different types of sensors.
        void setupAll();
        //Wrapper function for reading from each of the initialized
        //sensors.
        void runAll();
    //Function needed for functionality of the wireless programming.
void printUpload();

// The private function declarations that are used
// internally by the library. For Details on each of these
// functions see the source SmartFarmMeasure.cpp
private:
    String getDevAddress(DeviceAddress deviceAddress);
    void printData(DeviceAddress deviceAddress);
    void takeDecMeasurement(char i);
    void printBufferToScreen();
    boolean checkActive(char i);
    boolean setTaken(byte i);
    boolean setVacant(byte i);
    boolean isTaken(byte i);
    char printInfo(char i);
    byte charToDec(char i);
    char decToChar(byte i);
    void delayMilliseconds(int x);
    void finishUp();
};

#endif
Appendix B. SmartFarmfile_watcher.py to observe the incoming file from the WordPress site

#!/usr/bin/env python

...

SmartFarmfile_watcher.py

This script watches the watch_path directory for an incoming .ino file (See ... for other file types) When it detects that an .ino file was uploaded via the website, it moves the previous ino file that was uploaded to the board into the archive_path. It then moves the file uploaded from the website into the currentDAQ_path directory and calls UPLOAD_CurrentDAQ.py to upload the program to the SmartFarm v5.4 board.

Created by Caleb Fink and Jeremy Kerfs: May 2017
latest revision: June 11, 2017

...

# imports of required libraries for this script
import sys
import urllib2
import os
import subprocess
import glob
import shutil
from subprocess import Popen
from shutil import move
from time import sleep
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

# uploaded programs directory from wordpress.org
watch_path = "/var/www/html/wp-content/uploads/SmartFarmPrograms/"
# platformio project source file directory
currentDAQ_path = "/home/pi/SmartFarm_CurrentDAQ/src"
# archive directory of previously uploaded program
archive_path = "/home/pi/SmartFarm_Archives/"

"""this portion is run once the watcher/observer notices a file in the watch_path.
It does 2 file moves and then calls UPLOAD_CurrentDAQ.py which uploads the program to the board"""

class InoFileEventHandler(FileSystemEventHandler):
    def on_created(self, event):
        # stop ThingSpeakCode.py script completely
        ThingSpeakCode.kill()
print '\n{0:*^80}'.format(' Ending ThingSpeakCode.py script ')
# move the current DAQ program into the archives from CurrentDAQ directory
for p in glob.glob(os.path.join(currentDAQ_path, "*.ino")):
    f = os.path.split(p)[-1] #just the filename
    print ">Last uploaded program from SmartFarmBaseStation Website: " + f
    shutil.move(p, os.path.join(archive_path, f))
    print ">Archiving previous program " + f + " to archives directory: " + archive_path
# move newly uploaded file from watch directory to CurrentDAQ
for p in glob.glob(os.path.join(watch_path, "*.ino")):
    f = os.path.split(p)[-1] #just the filename
    shutil.move(p, os.path.join(currentDAQ_path, f))
    print ">New program to upload " + f + " is now located in:" + currentDAQ_path
print '\n{0:*^80}'.format(' Ending SmartFarmfile_watcher.py script ')  
# call and run the UPLOAD_CurrentDAQ script
p = subprocess.Popen(['python', "/home/pi/SmartFarm_CurrentDAQ/UPLOAD_CurrentDAQ.py"])
# close this watcher script completely so that it cn be restarted after upload
os._exit(-1)
if __name__ == "__main__":
    # this is the start of this script
    print '\n{0:*^80}'.format(' Running SmartFarmfile_watcher.py script ')
    # naming the event handler
    event_handler = InoFileEventHandler()
    observer = Observer()
    # watching/observing the watch_path directory
    observer.schedule(event_handler, watch_path, recursive=False)
    observer.start()
    print ">Observer is set to watch: " + watch_path
    try:
        # print ">Calling ThingSpeakCode.py script..."
        ThingSpeakCode = subprocess.Popen(['"python", "/home/pi/SmartFarm_CurrentDAQ/ThingSpeakCode.py"])
        print ">Calling Arduino_Serial.py script..." 
        ThingSpeakCode = subprocess.Popen(['"python", "/home/pi/SmartFarm_CurrentDAQ/Arduino_Serial.py"])
        while True:
            # continuous loop while ThingSpeakCode.py is running
sleep(2)
except KeyboardInterrupt:
observer.stop()
observer.join()
os._exit(-1)
Appendix C. ThingSpeakCode.py posts data to ThingSpeak.com

```python
#!/usr/bin/env python

""
ThingSpeakCode.py

>>>>>>Take note of myAPI<<<<<<

This program posts data to ThingSpeak.com a website hosting data from IoT devices. An account can be created at https://thingspeak.com/
To post data to the web, this script reads the serial data from the SmartFarm board, separates it, and posts it to the proper field in ThingSpeak. The Serial data format may be changed in .ino file uploaded to the SmartFarm board to better separate data here, or the formatting in this script can be changed to separate the data and post it. This script is an example to post Watermark sensor data.

Serial Data from the SmartFarm board comes in the following format:
---------------------------------------------------------------
1|Finding temperature sensors: Found 0 devices.
   |
2|Scanning Decagon 5TE addresses.
   |
3|No DECAGON 5TE sensors found.
   |
4|Found 0 Decagon 5TE sensors.
   |
5|Reading Watermark sensors...
   |
6|NA 125.27 NA 74.05
   |
7|Reading temperature sensors...
   |
8|Reading Decagon 5TE Sensors...
   |
9|Format: (Time elapsed), (Sensor ID, Type, and Version). (Measurements)
   |
10|Upload
---------------------------------------------------------------

Watermark measurement and posting to ThingSpeak. The two lines this script posts are these shown below which are 5 and 6 as shown in the format above
5|Reading Watermark sensors...
   |
6|NA 125.27 NA 74.05

To post this data, it has to first be read in, assigned a variable, and posted.
```
The script below is commented for explanation.

Created by Caleb Fink: May 2017
latest revision: June 15, 2017

```
# imports of required libraries for this script
import serial
import sys
import os
from time import sleep
import urllib2

# define an empty list
temp_list = []
# placement in the list reading
FirstWatermarkReadingPlacekeeper = 23
# the last serial data to trigger post to ThingSpeak
EndOfSerial = "Upload"

#Setup API and delay
myAPIKey = "yourAPIkeyhere"
ThingSpeakPostLimit = 15 #seconds between posting data
serialPort = '/dev/ttyUSB0' # port the USB XBee connects to
serialBaudRate = 57600 # baud rate of XBees

# setting up the serial port for the XBee and SmartFarm
SmartFarmSerialData = serial.Serial(serialPort,serialBaudRate)
SmartFarmSerialData.flushInput()
SmartFarmSerialData.flushOutput()
print '
{0:*^80}'.format(' Running ThingSpeakCode.py script ')
baseURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPIKey
print ">Data will be sent to: " + baseURL
try:
    while True:
        # read in serial data from the XBee
        words = SmartFarmSerialData.readline()
        # creates a list (trigger) that splits up (words/serial data) by the (" ")
        trigger = words.split(" ")
        # collects all serial data into a single list such as [0,1,2,3,4,5...] to parse out which sensors are
        which data
        temp_list.extend(trigger)
        # after collecting data it will then post it. The if statement doesn't execute until all serial data has
        finished collecting
        if trigger[0] == EndOfSerial:
            # post data only after it has finished collecting it
            WaterMark1 = temp_list [FirstWatermarkReadingPlacekeeper]
            print WaterMark1
            WaterMark2 = temp_list [24]
            print WaterMark2
```
WaterMark3 = temp_list[25]
print WaterMark3
WaterMark4 = temp_list[26]
print WaterMark4
# checks if the reading is a number and not NA
if WaterMark1.isdigit():
    print "%%d" % (WaterMark1)
if WaterMark2.isdigit():
    print "%%d" % (WaterMark2)
if WaterMark3.isdigit():
    print "%%d" % (WaterMark3)
if WaterMark4.isdigit():
    print "%%d" % (WaterMark4)
# post the data to ThingSpeak in fields 1 through 4
f = urllib2.urlopen(baseURL + "&field1=%s&field2=%s&field3=%s&field4=%s" % (WaterMark1, WaterMark2, WaterMark3, WaterMark4))
    # print f.read()
f.close()
    # ThingSpeak posting limit of 15 seconds
sleep(int(ThingSpeakPostLimit))
except KeyboardInterrupt:
    print 'exiting.'
    os._exit(-1)

---

**Appendix D.** *UPLOAD_CurrentDAQ.py calls compile and upload using PlatformIO*

```
#!/usr/bin/env python

""
UPLOAD_CurrentDAQ.py

This script cleans up the previous programs' build files, compiles the new program, and waits until the SmartFarm board indicates it is awake and ready to be programmed. After indication of being awake, the new program is uploaded to the SmartFarm board. The indication of when the SmartFarm board is awake is determined by the serial data and the trigger of 'Upload' from the SmartFarm board. When the trigger is caught, platformIO "an open source ecosystem for IOT development" is used to upload to the SmartFarm board.

More on PlatformIO can be found at http://platformio.org/

Created by Caleb Fink March 2017
latest revision: June 15, 2017
"""
# imports of required libraries for this script
import serial
import time
import os
import subprocess
from subprocess import call

# heading for this python script
print '{0:*^80}'.format(' Running UPLOAD_CurrentDAQ.py script ')
print 'Will upload to SmartFarm Board on next wakeup'
# cleanup the object files from last build
print 'Cleanup from last upload'
call(['platformio', 'run', '-t', 'clean'])
# compile the program before uploading
print 'Compiling'
call(['platformio', 'run'])
# setting up the serial port for the XBee and SmartFarm
SmartFarmSerialData = serial.Serial('/dev/ttyUSB0', 57600)
line = None
# clearing the serial buffer
SmartFarmSerialData.flushInput()
SmartFarmSerialData.flushOutput()
try:
    print 'Waiting for upload request from smartfarm board...'
    while True:
        read the serial data from the SmartFarm board by lines
        line = SmartFarmSerialData.readline()
        trigger = line
        # splits the words in the line by a space
        trigger = line.split(' ')  
        # when the word Upload is found from the SmartFarm measurement data, this triggers the
        upload of the .ino file
        if trigger[0] == 'Upload':
            print 'Request received...upload in progress'
            # upload compiled .ino program to SmartFarm board
            call(['platformio', 'run', '-t', 'upload'])
            print 'Returning to measurement mode...'
            # clearing the serial buffer
            SmartFarmSerialData.flushInput()
            SmartFarmSerialData.flushOutput()
            # close the serial port before proceeding out of this script
            SmartFarmSerialData.close()
            print '{0:*^80}'.format(' Ending UPLOAD_CurrentDAQ.py script ')
            # after upload, the watcher and thingspeak scripts start again
            p = subprocess.Popen(['python', 
'/home/pi/SmartFarm_CurrentDAQ/SmartFarmfile_watcher.py'])
            os._exit(-1)
except KeyboardInterrupt:

print "Script Canceled"
**Appendix E. Useful Raspberry Pi commands**

<table>
<thead>
<tr>
<th>Useful Raspberry Pi Terminal Commands:</th>
</tr>
</thead>
</table>

Installing applications over internet:
sudo apt-get install <name of software>

Open folder/change directory:
cd folder_name
cd /opt/vc/src/hello_pi

Folder back /Change directory back:
cd ..

Remove file from directory:
rm this_file.txt /someDirectory/AnotherDirectoryinSomeDirectory

Move a file in the current directory to another directory:
mv this_file.txt /someDirectory/ThisDirectory

Make a directory/folder:
mkdir /someDirectory/NewDirectory

Run python script:
sudo python example.py

Find location of usb device (plug in, then remove to see what changed in the list):
ls /dev/tty*

Upload Smartfarm board using Makefile:
make upload

Shutdown Raspberry Pi:
sudo poweroff