5-2019

Design of Remote Datalogger Connection and Live Data Tweeting System

Zachary Wofford

Design of Remote Datalogger Connection and Live Data Tweeting System

Zachary Wofford

Biological Engineering Program

Biological and Agricultural Engineering Department

College of Engineering

University of Arkansas

Undergraduate Honors Thesis

PROJECT SUMMARY

Low-Impact Development (LID) is an attempt to sustainably respond to the potential hazards posed by urban expansion. Green roofs are an example of LID design meant to reduce the amount of runoff from storm events that are becoming more intense and less predictable while also providing insulation to buildings. LID has not yet been widely adopted as it is often a more expensive alternative to conventional infrastructure (Bowman et. al., 2009). However, its benefits are apparent. The University of Arkansas Honors College awarded a grant to research the large green roof atop Hillside Auditorium. One part of this grant is aimed at educating the public on the benefits LID infrastructure and encourage its development. To accomplish this task, a Raspberry Pi was programmed to operate in tandem with a Campbell Scientific CR1000 datalogger to collect, organize and tweet data to the public under the moniker, "Rufus the Roof." It is believed that personifying the roof allows data to be conveyed in an entertaining manner that promotes education and public engagement in the LID design.

The Raspberry Pi was initially intended to collect data and publish tweets automatically on a live basis. However, automation was not realized due to time constraints and challenges in establishing connection to the datalogger. Instead, a system was developed that allowed the remote transfer of environmental data files from a datalogger on the green roof. Along with remote file transfer protocol, several Python scripts were written that enabled tweets to be published by the Raspberry Pi.

The design was successful. Manual remote file transfer and tweeting was achieved. Full automation remains to be achieved, but the Python scripts are built with the capability to operate automatically. The conditions are in place for future development of the project in order to achieve full autonomy. A fully automated system could open the doors for more widespread public engagement in the value and benefits of Low-Impact Development initiatives.

I. INTRODUCTION

Global average temperatures have increased alongside increases in population and consumption. These changes pose serious threats to the current standard of living as weather events become less predictable and more intense (Westra et. al., 2014). Along with this trend, increases in population are accompanied by corresponding increases in urbanization. Peak runoff is known to increase with urbanization due to the conversion of infiltrating soils to pavements and other impervious surfaces (Miguez et. al., 2015). Low-impact development (LID) is a strategy to sustainably overcome the stormwater runoff hazards posed by climate change and urbanization (PGCo, 1999). Green roofs, also commonly referred to as vegetative or living roofs, are representative of LID through their capture of stormwater and capacity to insulate. An Honors College research grant was awarded to members of the Biological Engineering Department with the intent to study the impacts of a green roof on the University of Arkansas campus. The site selected was the large, extensive green roof above Hillside Auditorium due to its high visibility to passing students. The ongoing study collects environmental data regarding the green roof via a network of sensors connected to a Campbell Scientific CR1000 datalogger (Image 1; Image 2). Along with examining the physical impacts of the green roof, the project also sought to educate the public on how LID designs can be worthwhile investments. As a complimentary activity this thesis seeks to design a method of transmitting collected environmental data to the public in an open and accessible way on a live basis.

On a study abroad course at the University of Gent, Belgium during my freshman year I encountered a tree with embedded sensors that collect and post biological data to Twitter on a live basis. The system struck me as an ingenious way of enabling public engagement in science. That tree also happens to be a member of a large network of trees across European universities that operate on Twitter under the name TreeWatch (Steppe et. al., 2016). The network was designed to study how trees are responding to climate change in their daily biological functions. It is believed that trees can serve as early warning signals to climate change through die offs (Neumann et. al, 2017). The TreeWatch network seeks to understand more about the trees' reactions to climate change before they die while also raising public awareness of the hazards posed by climate change. TreeWatch believes that tweeting sensor arrays have an intrinsic educational power (Steppe et. al., 2016). This project intends to utilize that power in order to have a lasting impact on the public's understanding of LIDs and care for sustainable initiatives.

Disseminating data to the public is the primary goal of this project. However, in the design process new insights were gained regarding the competitive hold scientific instrumentation companies place on their products and markets. Campbell Scientific offers a manual regarding connecting internet capable devices to the CR1000 datalogger. However, all of the options in their product manual refer to using compatible products sold by Campbell Scientific. This limits open access by researchers and imposes increased system costs. Frustration has been expressed in online forums where individuals are trying to establish connections without purchasing Campbell's compatible hardware.

One aspect of the design will be to program tweets coming from the green roof in first person. This is inspired by the TreeWatch network as a means of further engaging the public. By personifying the roof – giving it a name and a voice – it is hoped that the study will become more personable and impactful. Getting the public engaged in LID infrastructure research on its own is a likely a challenge. Offering the research in a fun and approachable manner may ease this challenge. The first step in personifying a non-sentient object is to name the object. It was decided that the roof would tweet under the name Rufus the Roof. The selected twitter handle for the account was @UAGreenRoof.

In considering how the datalogger would be able to connect to the internet and compile collected data into tweets the decision was made to use a Raspberry Pi microcomputer. This option was chosen over the various alternatives such as Arduino microprocessors and Campbell Scientific's own connection device due to its low price, high accessibility, and ease of use. The Biological Engineering department has several Raspberry Pis that are used in the Measurements and Controls lab, so one was readily available for use in this project. However, if this project were to be carried out by any individual without the privilege of lab access they would only see a cost of $35 for the device (not including monitor, keyboard, or mouse). Once set up and connected to a monitor, the Raspberry Pi behaves like any other computer. Thus, programming on it is a more intuitive process than on the desktop-lacking Arduino. The Raspberry Pi also comes with a built in Wi-Fi receiver that makes it perfect for on-campus remote studies.

II. LITERATURE REVIEW

Low-Impact Development (LID) is a relatively new concept in infrastructure design. Therefore, it is unlikely that the general public fully grasps the functionality of some designs (Bowman, et. al., 2009). This limited understanding extends to LID green roofs like that on Hillside Auditorium. Raising public awareness of LID systems is an important step in their promotion and development on a wider scale. The proposed live data tweeting system attempts to engage the public in the study of LID benefits. Extending findings to the community in an open and accessible manner is a form of citizen science.

The primary objective of the live data system is to educate the public about the benefits of LID green roofs. This community science initiative will not receive direct scientific contributions from the community. Rather, it will provide a means of engagement with an ongoing study through promotion. Promotion of the science will allow for a wide array of the public to better understand water-based LID through regular interaction (Mercer, 2018). An increased level of engagement could also potentially incite new scientific ideas from the general public.

The study also aims to exhibit the value of widespread livestream sensor systems. Specifically, the study aims to exhibit how these systems can be both affordable and effective. Some universities are considering the deployment of environmental sensing arrays in order to become a "smart campus" where students are given live information about occupancy of buildings and other environmental data. These studies have found that Raspberry Pi microcomputers are capable of accomplishing this at a low cost (Hentschel, 2016). Decreasing cost is essential to the design of large scale sensing systems. The cost of computing has decreased dramatically since the technology's inception. A Raspberry Pi processor is a credit card sized computer that is priced affordably at $35 USD. This affordability makes the widespread deployment of sensing systems a reasonable venture economically.

Widespread sensing is ultimately necessary as a detector of climate change issues. Biological systems are often our first sign of negative climatic change (Steppe, et. al., 2016). However, the way that they present their data is through wilting, discoloration, and eventually die-offs. Live remote sensing and aggregation of data allows us to detect the stressors that organisms are feeling in real time – as they are feeling them. Sensing does not prevent die offs or climate change as a whole. But it does allow for early warnings and a better understanding of the rate of change. Tweeting the data that is collected is a way of conveying the data to a broad audience so that they too can feel the change. Tweeting data is not absolutely crucial, the widespread audience likely cannot make direct use of the data, but larger quantities of people collectively witnessing change is important. The collective of people are much more acquainted with social media than scientific publications.

Twitter specifically possesses an ease and familiarity that is ultimately a fun way to process information. One example is the University of Arkansas Student Union. It has a twitter (@Arkansas Union) and uses it to assist in event planning and organization as well as student outreach. If a student union can have a personality that has an ultimate educational purpose, why can't a tree, or a green roof? It is believed that bringing it to life and giving it a voice for our campus (even if it simply provides data) will give students a better understanding of the sustainable development inherent in its design.

Specifically, it should be able to quantitatively show students how sustainable design is a good thing, and how it is impacting the environment we inhabit.

III. METHOD

The initial plans for the design involved automated data fetching, organization into graphs, and tweeting with an accompanying randomized message. The first step in designing the live tweeting system was to establish a stable and consistent connection between the Raspberry Pi microcomputer and the Campbell CR1000 datalogger. However, because of Campbell Scientific's aversion to third party compatibility, a great deal of time was spent merely establishing the connection from the Raspberry Pi to the datalogger. Because of this challenge and the accompanying difficulty in coding automated data fetching and organization programs the design was shifted to a remote file transfer device. The final design included a set protocol for connection and file transfer from the datalogger as well as a series of Python scripts that allow for convenient dissemination of information regarding the green roof. The overall connection protocol and series of commands necessary for the remote publication of a tweet from the green roof is summarized in Figure 1 below.

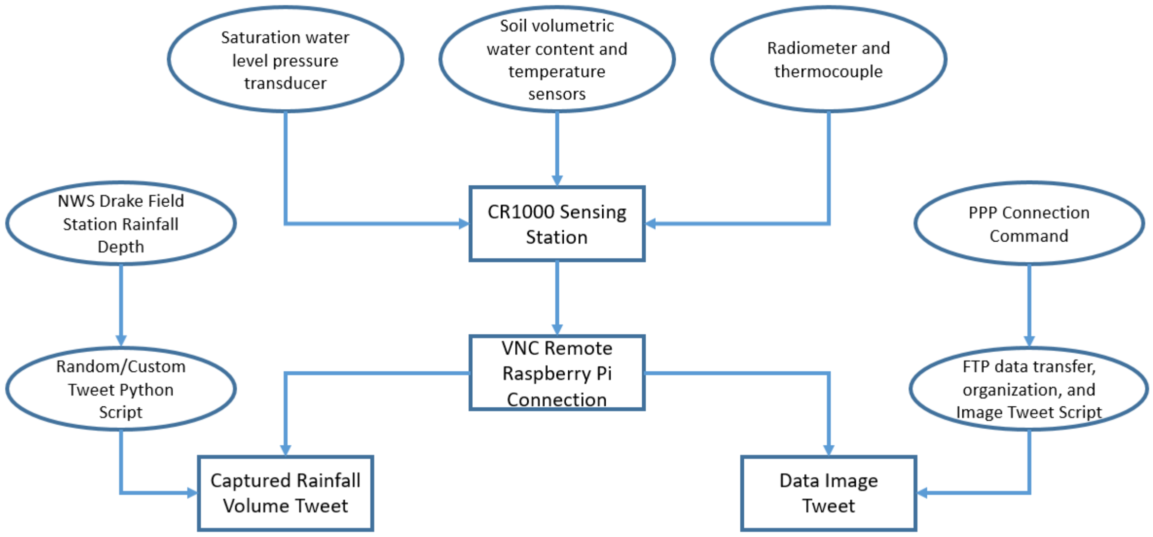

Saturation water level pressure transducer
Soil volumetric water content and temperature sensors
Radiometer and thermocouple
NWS Drake Field Station Rainfall Depth
CR1000 Sensing Station
PPP Connection Command
Random/Custom Tweet Python Script
VNC Remote Raspberry Pi Connection
FTP data transfer, organization, and Image Tweet Script
Captured Rainfall Volume Tweet
Data Image Tweet

Figure 1: Protocol diagram for remote tweet publication

1. **Raspberry Pi Setup:**

Raspberry Pis require a 5v 2.5A power supply, keyboard, mouse, and monitor for initial setup. The Linux-based operating system, Raspbian, is loaded manually onto a blank Pi via a micro-SD card. The Linux-based operating system is automatically installed upon the first start up. After that, the computer and its pre-programmed software, Raspbian, is ready to use. The first step in making the device remotely accessible was establishing it as a Virtual Network Computing (VNC) server. VNC is a free software that allows any computer with approved credentials to access remote servers and operate them as a normal desktop. If the initial design plans of full automation were realized, a VNC would be unnecessary. However, because the design shifted to manual operations it became a key component. Logging on to the VNC software from a remote location is the first step in fetching data and operating tweeting scripts.

It was particularly difficult to provide power to the Raspberry Pi from the CR1000. The CR1000 offers multiple ports for powering external devices including two 12V ports, one switching 12V, and a 5V port. Because the Raspberry Pi requires 5V the initial decision was to use the 5V port. However, the CR1000 does not provide enough current through this port. The Raspberry Pi requires upwards of 2.5A while the CR1000 was shown with a multimeter to only produce around 250mA. This low power output made the 5V port ineffective. When examining the CR1000 operation manual it was noticed that the 12V ports have an accompanying amperage of up to 3A. Supplying 12V to the Raspberry Pi could cause serious damage to the circuitry, however it is possible to step down the voltage from 12V to 5V while maintaining the needed current. A linear voltage regulator was tested as the solution to this problem. Linear voltage regulators take an input of 12V and provides an output of 5V. However, during testing the small device reached an extremely high temperature, slightly melted, and then caused the Raspberry Pi to lose power. The high temperature reached was due to 7V and 3A (21W) being lost as heat. A more novel solution was found through the dismantlement of a 12V to 5V USB car charger intended for use in cigarette lighter ports (Image 3; Image 4). Inside these devices one can find a positive and negative terminal for connection. These devices are designed in such a way as to not produce excess heat. So, a 12V feed from the CR1000 was attached to the positive terminal (spring) of the car charger while another wire connected from the negative terminal to the "ground" port of the CR1000 (Image 5).

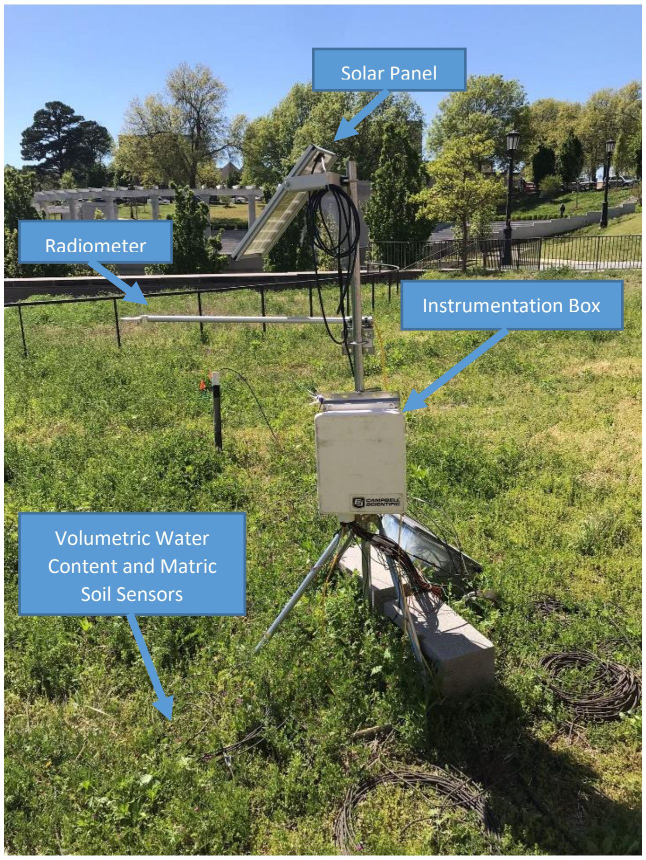


Image 1: Sensing station on Hillside Auditorium Green Roof

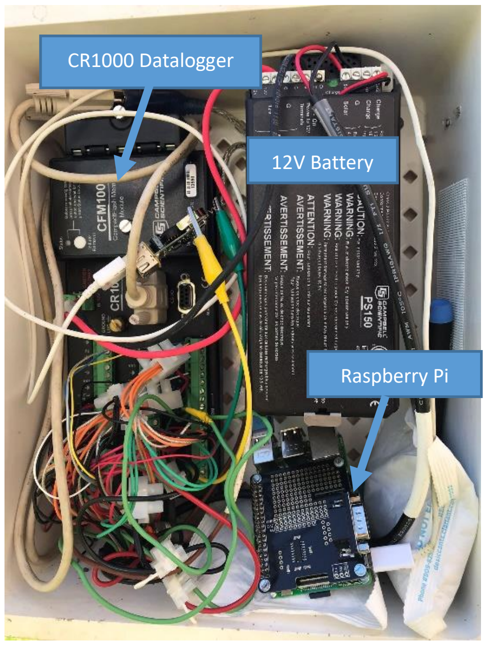


Image 2: Instrumentation inside sensing station

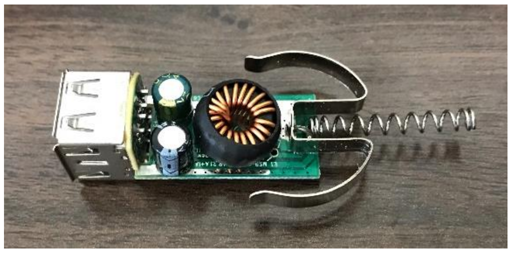


Image 3: 12V to 5V USB car charger with casing removed

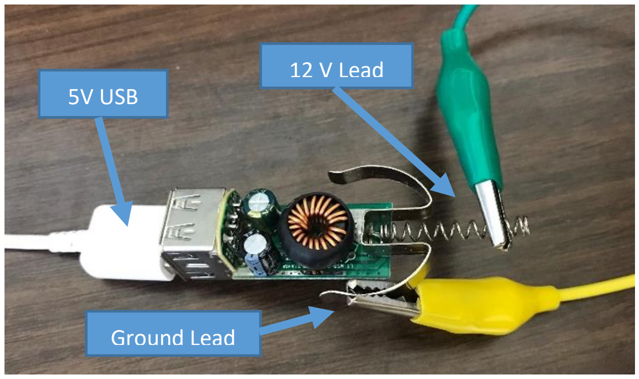


Image 4: Voltage reducer with lead wires and USB assembled

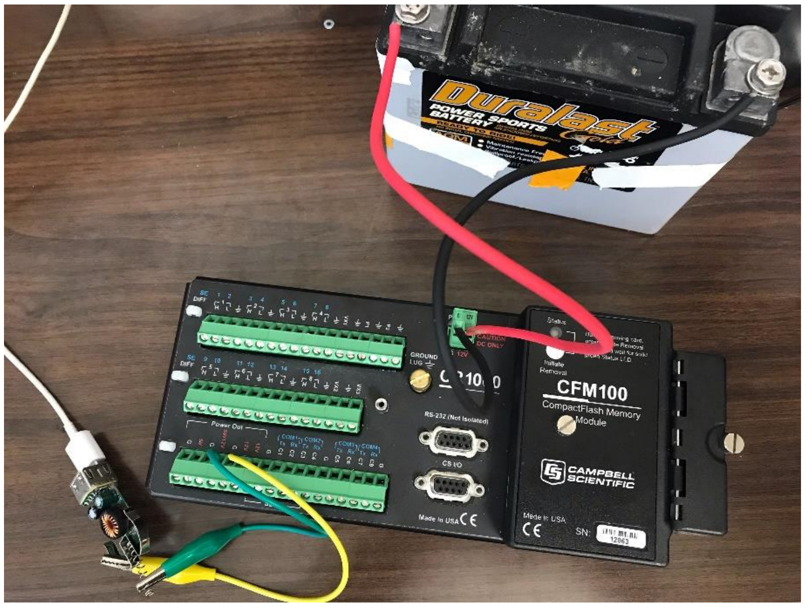
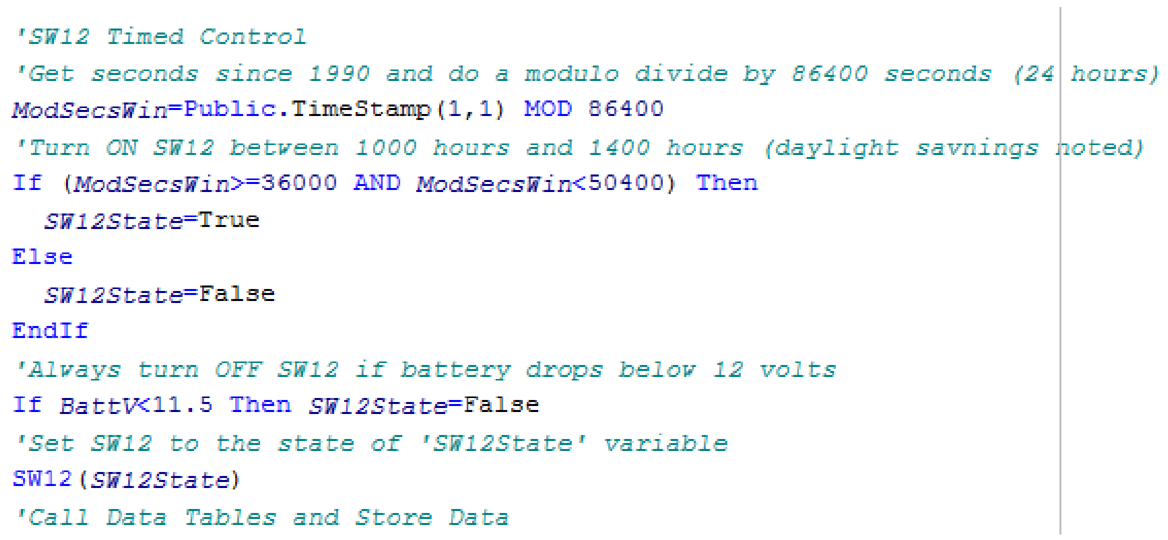


Image 5: Power adapter connected CR1000 and battery via switching 12V port

The Raspberry Pi requires significantly more power to operate than the environmental sensors attached to the CR1000. In order to avoid the overuse of battery power supplied by the solar panel on the sensing station it is necessary to power down the Raspberry Pi when not being used. Luckily, the CR1000 comes equipped with a “switching 12V” port that can be controlled by the operating script of the datalogger, CRBasic. This addition is simple and can be adjusted to only have the Raspberry Pi powered on when needed for file transfer. The switching 12V port was programmed to only be active from 11:00am to 4:00 pm every day via the CRBasic code in Figure 2 below.



```
'SW12 Timed Control
'Get seconds since 1990 and do a modulo divide by 86400 seconds (24 hours)
ModSecsWin=Public.TimeStamp(1,1) MOD 86400
'Turn ON SW12 between 1000 hours and 1400 hours (daylight savnings noted)
If (ModSecsWin>=36000 AND ModSecsWin<50400) Then
  SW12State=True
Else
  SW12State=False
EndIf
'Always turn OFF SW12 if battery drops below 12 volts
If BattV<11.5 Then SW12State=False
'Set SW12 to the state of 'SW12State' variable
SW12(SW12State)
'Call Data Tables and Store Data
```

Figure 2: Switching 12V power CRBasic code

One challenge with the switching 12V power supply is that the Raspberry Pi requires a proper shut down before power is disconnected in order to avoid corruption of the operating system. This process was accomplished with a program built in to the Raspberry Pi called "crontab." Crontab allows for terminal commands to be scheduled based on the Raspberry Pi's onboard clock. To schedule a shutdown for 3:50pm with crontab the following script must be added to the "/etc/crontab" directory:

```
50  15        *  *  *        root      shutdown
```

2. CR1000 Connection:

Once the Raspberry Pi and its VNC software are functional a connection to the CR1000 datalogger must be established. This connection proved to be the most difficult portion of the project. The CR1000 connects to computers via an RS-232 cable. The Raspberry Pi has no RS-232 ports. Luckily, the device is built to be manipulated to accomplish a wide variety of tasks. The first attempted solution was installing an RS-232 adapter. This solution came in the form of a HAT (hardware on top) that can be soldered to the Raspberry Pi's pins. This device showed no promise, however, and any attempts at establishing a connection failed. The cause for its ineffectiveness is unknown. As an alternative, it was decided that a USB to RS-232 adapter would be most effective. This adapter was included in the final design.

This project was not the first time someone has attempted to connect a Raspberry Pi – or any other computer – to a CR1000 datalogger. Therefore a wide array of predeveloped code exists online from sources such as Github.com. However these sources consistently failed to establish steady connections, likely due to being out of date and incompatible with current Python and Linux software versions. The creators of the source code were unresponsive to any troubleshooting requests.

At this point in the project I teamed up with Biological Engineering PhD student, Colby Reavis, to establish a stable connection. Colby and I tried more incompatible, predeveloped code before deciding to establish the connection ourselves. Colby suggested a connection protocol I was unaware of called Point-to-Point Protocol (PPP). The CR1000 datalogger has PPP imbedded into its design. We simply had to assign an IP address to the datalogger and manually tell the Raspberry Pi, via a PPP command, to establish a link between its own IP address and the one assigned to the datalogger. The command is as follows:

```
sudo pppd /dev/ttyUSB0 115200 debug noauth nodetach
192.168.27.236:10.0.0.1
```

- `sudo` – tells the Raspberry Pi that the command is given full administrator privileges
- `pppd` – initiates the Point-to-Point Protocol command
- `/dev/ttyUSB0` – tells the pppd command to use the USB port that is being used by the USB to RS-232 adapter
- `115200` – the baud rate (communication speed) of the CR1000
- `debug noauth nodetach` – suggested PPP alterations to ensure stable connections and bypass authorization requirements

- `192.168.27.236:10.0.0.1` – the Raspberry Pi IP address and the assigned CR1000 IP address

Upon the input of this command the two devices are effectively communicating. Communication can be confirmed by pinging the datalogger with the following command:

```
ping 10.0.0.1
```

If the Raspberry Pi control terminal shows ping communication then the two devices have successfully been connected.

3. **Twitter Application:**

Twitter has a section of their website that serves as a platform for application developers. In order to have automated scripts write and publish tweets it was first necessary to register the design as an application on the platform. Once the profile for @UAGreenRoof was operational an application request was sent to Twitter under the name "UAGreenRoofBot" in order to acquire access to the developer platform. The request was immediately accepted and access was made available to important information required by the Python scripts. In order to bypass the typical steps of posting a tweet such as entering a password and logging in, the Twitter developer platform grants access to a profile's specific tokens and keys that allow remote access from external programs. These tokens and keys are basically long strings of letters and numbers that serve as a username and password that grant login bypass access to external programs. With this information in hand, the next step was to create a module in Python that saved these tokens and keys so that other Python modules can simply pull from them in order to achieve access to the account. The module was titled "auth.py" and looked like the script below (tokens and keys redacted to maintain privacy):

```
consumer_key = 'insert_key'
```

```
consumer_secret = 'insert_secret'

access_token = 'insert_token’

access_token_secret = 'insert_token_secret'
```

4. **Python Script:**

With “auth.py” saved it was able to be conveniently referenced by any new Python script. Because full automation was not reached in this design, several different scripts were written for different occasions in order to make the device easily accessible by any future students who might take up the mantle of maintaining the Twitter account. These Python scripts actually operate independent of the datalogger and are built as a way to easily publish graphs, captured water volume data, and randomized messages regarding water capture. The initial design had these scripts pulling data from the datalogger, designing graphs of daily soil water balances, and posting them as images. However, this objective proved too complex of a process for the time available so it was scrapped in favor of simple volumes of captured rain calculations. Below is an example of the Python script titled “Rain2Volume_random.py”:

```python
rainfall_depthcm = 1.27      #Rainfall data from NWS Drake Field Station
rainfall_depthm = rainfall_depthcm * 0.01    #Conversion to depth in meters
GR_area = 909                #Python Module that enables output values to be rounded

if rainfall_depthm > 0:
    captured_volume = rainfall_depthm*GR_area
    output = round(captured_volume,2)
    print(output)
else:
    print('No Rain.') #This component is built in for any future automation initiatives

import random                      #Python module for randomization of strings

caption = ["Yesterday was a rainy one! I captured {} cubic meters of water.",
           "If I were just a regular ole roof, {} cubic meters of rain would have run off into municipal stormwater systems yesterday.",
           "Instead of contributing to potential flooding and erosion, I captured {} cubic meters of yesterday's rain. My plants are loving it!",
           "I captured {} cubic meters of water! It's going to hang out in my soil for a while. But eventually my plants will send it back to the atmosphere via evapotranspiration!",
           "Hey Hillside Auditorium students! Not only is there a vibrant plant and soil ecosystem above your heads, there's also {} cubic meters of water from yesterday's rain event!"
]
random_caption = random.choice(caption) #Selects random string from a list of strings

from twython import Twython        #Python module built for communication with twitter

from auth import (consumer_key,
                  consumer_secret,
                  access_token,
                  access_token_secret
                  )

twitter = Twython(consumer_key,
                  consumer_secret,
                  access_token,
                  access_token_secret
                  )

message = random_caption.format(output)
twitter.update_status(status=message)
print("Tweeted: %s" % message)
```

Figure 3: Python script for random captured rainfall tweet

This script operates by calculating a captured volume based on rainfall depth, rounding it to two decimal places, selecting a random caption to be tweeted via a list of strings, inputting the calculated value into the empty braces of the randomly selected caption via the command “random_caption.format(output),” and posting the message value to twitter. The “.format(output) command takes any “{}” and replaces it with an assigned value. The script is primarily used for quickly calculating and publishing a randomized tweet the day following a precipitous day.

The following script, “Rain2Volume_nonrandom.py” is used when a tweet requires more customization:

```python
rainfall_depthcm = 3.79                   #Rainfall data from NWS Drake Field Station
rainfall_depthm = rainfall_depthcm * 0.01 #Conversion to depth in meters
GR_area = 909                             #Area of the Green Roof in square meters
from decimal import Decimal     #Python Module that enables output values to be rounded

if rainfall_depthm > 0:
    captured_volume = rainfall_depthm*GR_area
    output = round(captured_volume,2)
    print(output)
else:
    print('No Rain.')#This component is built in for any future automation initiatives

from twython import Twython    #Python module built for communication with twitter
from auth import (consumer_key,
                  consumer_secret,
                  access_token,
                  access_token_secret
                  )

twitter = Twython(consumer_key,
                  consumer_secret,
                  access_token,
                  access_token_secret
                  )

message = "The last couple days have been rainy! I've captured {} cubic meters of water and helped reduce the load on Fayetteville's stormwater management systems!".format(output)
twitter.update_status(status=message)
print("Tweeted: %s" % message)
```

Figure 4: Python script for custom captured rainfall tweet

This script operates like “Rain2Volume_random.py” except it foregoes randomized messages and allows more control. In the above example it was used to give the captured volume total for a series of rainy days.

The above Python scripts act independently of the green roof datalogger and use rainfall data generated by the National Weather Service rain gauge at Drake Field. They are used to educate the public on the primary function of green roofs which is to decrease impervious areas in a watershed and reduce the amount of runoff from a precipitation event.

The final Python script is a simple program that uploads an image file to twitter with an accompanying caption. The example, “Image_tweet.py” is displayed below:

```python
from twython import Twython
from auth import (consumer_key,
                  consumer_secret,
                  access_token,
                  access_token_secret
                  )

twitter = Twython(consumer_key,
                  consumer_secret,
                  access_token,
                  access_token_secret
                  )

message = "insert_image_caption"
image = open('Image_file.jpg', 'rb')
response = twitter.upload_media(media=image)
media_id = [response['media_id']]
twitter.update_status(status=message, media_ids=media_id)
print("Tweeted: %s" % message)
```

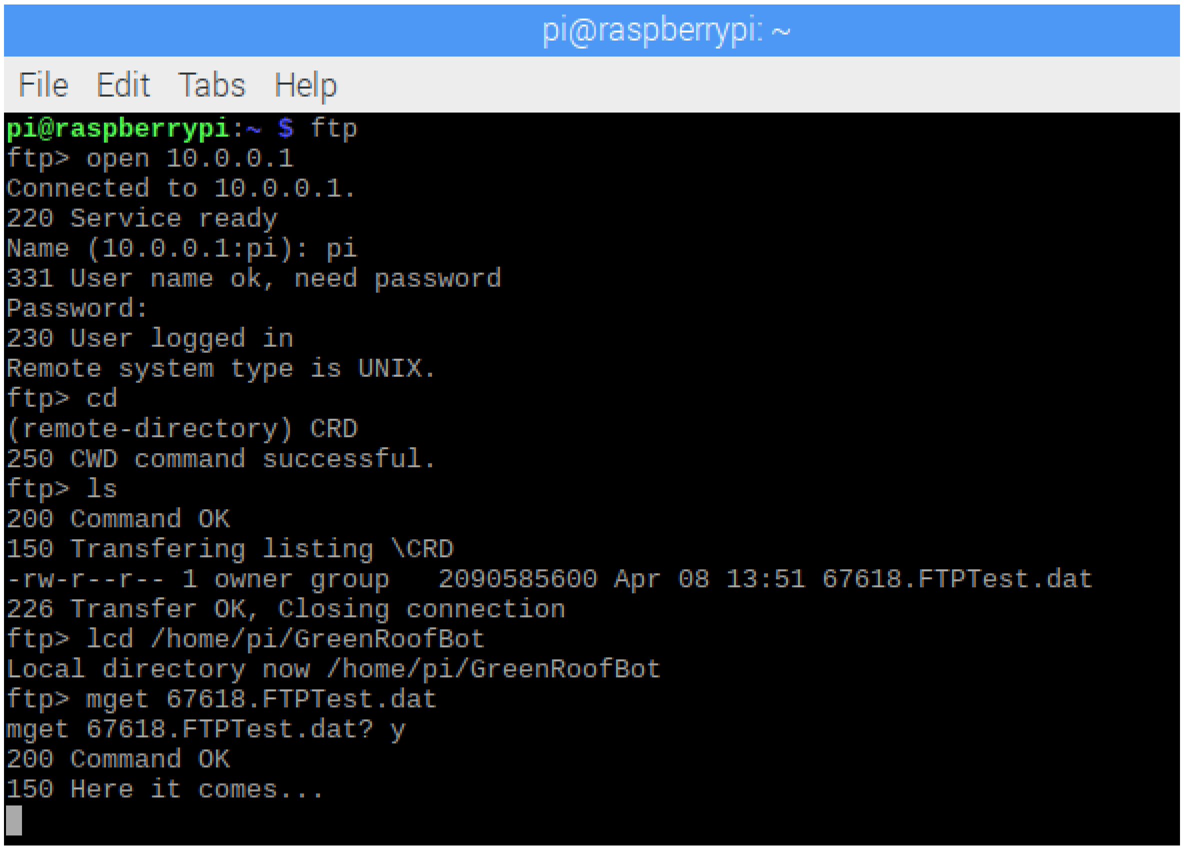
Figure 5: Python script for custom image tweet

The function of this script could just as easily by replicated manually on the Twitter app or website without any complicated Python syntax. However, it was designed with an automated system in mind. One that would create the image file from data fetched from the datalogger and post it on a daily basis to update and educate the public on the environmental conditions of the green roof. The remaining challenge here was not automated tweeting, but automated fetching and organization of data. Data fetching was achieved on a manual level via a 47 year old program called “File Transfer Protocol.”

5. File Transfer Protocol:

The most significant breakthrough of this project was the successful transfer of a file from the data logger to the Raspberry Pi. The significance is because remote file transfer became the primary goal of the design after the pivot was made from automation. File Transfer Protocol (FTP) is an ancient method of sending and receiving files from two devices over any stable connection. For this project it was first successfully tested via a Graphical User Interface (GUI) called Filezilla. Filezilla allows the user to input the IP address and port of the server (in this case the CR1000) to which the Raspberry Pi is connected. After that, the program runs an FTP and allows the free transfer of files between the two devices.

Once it was known that FTP was effective and file transfer was possible, operating FTP from the Raspberry Pi command terminal was faster and simpler than using Filezilla. The Linux commands and responses for FTP are as follows:



```
pi@raspberrypi:~ $ ftp
ftp> open 10.0.0.1
Connected to 10.0.0.1.
220 Service ready
Name (10.0.0.1:pi): pi
331 User name ok, need password
Password:
230 User logged in
Remote system type is UNIX.
ftp> cd
(remote-directory) CRD
250 CWD command successful.
ftp> ls
200 Command OK
150 Transfering listing \CRD
-rw-r--r-- 1 owner group   2090585600 Apr 08 13:51 67618.FTPTest.dat
226 Transfer OK, Closing connection
ftp> lcd /home/pi/GreenRoofBot
Local directory now /home/pi/GreenRoofBot
ftp> mget 67618.FTPTest.dat
mget 67618.FTPTest.dat? y
200 Command OK
150 Here it comes...
```

Figure 6: File Transfer Protocol (FTP) command example

- `ftp` – starts the FTP program
- `open 10.0.0.1` – establishes a connection between the Raspberry Pi and the CR1000

- Enter username and password of the FTP server
- `cd, CRD` – changes the directory in which files are being pulled from to the CR1000 memory card
- `lcd /home/pi/GreenRoofBot` – sets the location on the Raspberry Pi for the files to transfer to
- `mget` – command to download files (must confirm with `y/n`)

6. **Operation Steps:**

This project is designed to continue past the date of thesis publication because the honors research project it accompanies will continue through the 2019 calendar year. It seems apparent that the operation of this design must be made accessible to any future students who wish to use it. It is also apparent that this project is not alone in the quest to establish consistent and stable connection between a Raspberry Pi and a CR1000. I will therefore detail the steps necessary to operate the system below:

File Transfer and CR1000 data tweet:

1. Log into VNC viewer on your personal computer
2. Log into the VNC server of the Raspberry Pi
3. Run the PPP command in the Raspberry Pi command terminal using the script in Figure 5.
4. Establish an FTP connection
5. Transfer needed data files from the CR1000 to the Raspberry Pi
6. Open the ".dat" data file as a CSV file delimited by spaces and create a graph of the data
7. Save the graph as an .jpg image
8. Open the Image_tweet.py module from the folder titled "GreenRoofBot," craft a caption to accompany the data, and direct the script to open and upload the .jpg image

9. Run the Python script

Volume of water captured tweet:

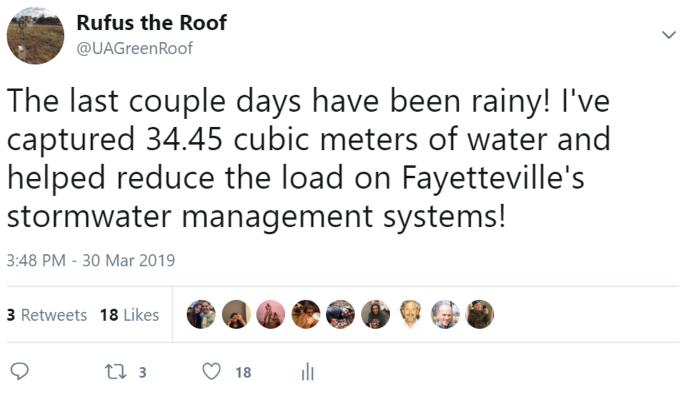
1. If rain only occurred the previous day
    a. Open the Rain2Volume_random.py module from the folder titled "GreenRoofBot"
    b. Find total rainfall (cm) from the NWS page for the Drake Field weather station
    c. Input the total depth of rain (cm) on the rainfall_depthcm value
    d. Run the Python script
2. If rain occurred over a series of days or if attempting to send a customized tweet
    a. Open the Rain2Volume_nonrandom.py module from the folder titled "GreenRoofBot"
    b. Find total rainfall (cm) from the NWS page for the Drake Field weather station
    c. Input the total depth of rain (cm) on the rainfall_depthcm value
    d. Write accompanying tweet with "{}" as a placeholder for the total volume value
    e. Run the Python script

IV. RESULTS

If a stable connection is in place and any potential troubleshooting complete, following the file transfer operation steps in section III.6 will deliver a successful remote data collection. This was proven multiple times to be a successful procedure. Following the procedure of randomized or customized tweeting was also shown to be consistently effective. While an automated data collection, organization, and tweeting system was not successfully realized, the structure is in place for its establishment. All of the operational Python script was designed to be built upon for the realization of full automation. The possible procedure for continuation of this design is discussed in the Discussion and Future Opportunities section.

1. **Captured Rainfall Volume Tweets:**

This format for the Twitter account was inspired by the daily updates of sap flow delivered by TreeWatch. Periodically collecting data from the green roof sensing station and manually organizing it into graphs to tweet is more time consuming for both the account curator and the account followers. A simple, regular, and significant data point from the green roof was sought after in order to emulate TreeWatch's daily sap flow tweets. One of the most significant benefits of green roofs are their abilities to capture water that, on a typical roof, would be considered runoff from an impervious surface. This feature of green roofs is the simplest to convey to the public. It avoids complicated water or radiation balances while promoting the primary function of green roofs. Below I display both random and custom tweets regarding captured rainfall (Image 6; Image7):


Rufus the Roof
@UAGreenRoof

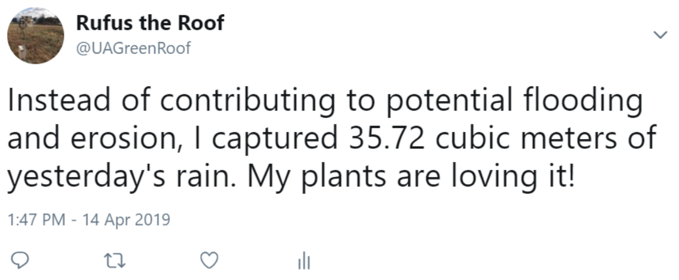
The last couple days have been rainy! I've captured 34.45 cubic meters of water and helped reduce the load on Fayetteville's stormwater management systems!

3:48 PM - 30 Mar 2019

3 Retweets 18 Likes

3 18

Image 6: Custom captured rainfall volume tweet


Rufus the Roof
@UAGreenRoof

Instead of contributing to potential flooding and erosion, I captured 35.72 cubic meters of yesterday's rain. My plants are loving it!

1:47 PM - 14 Apr 2019

Image 7: Random captured rainfall volume tweet

These tweets were generated based on rainfall data retrieved manually from the National Weather Service station at Drake Field.

2. Data Tweets:

In order to prevent monotony and utilize real environmental data collected on site the system was also designed to tweet visual data in the form of graphs. This is a deviation from TreeWatch and showcases the wide array of sensors that are connected to the system. Below are examples of graphs that were collected, compiled, and tweeted by the Raspberry Pi (Image 8; Image 9):

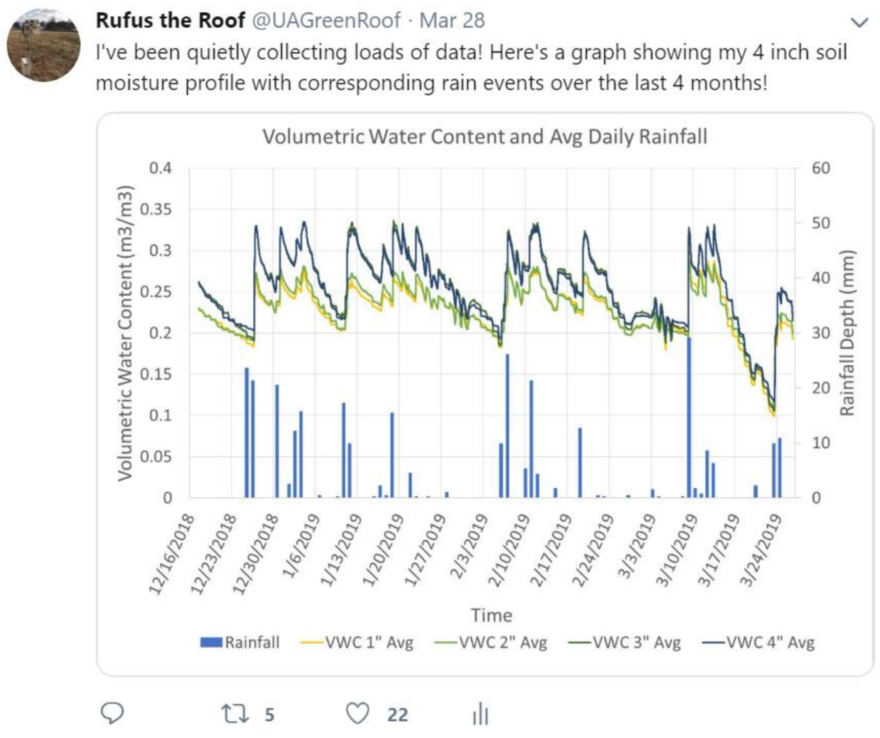
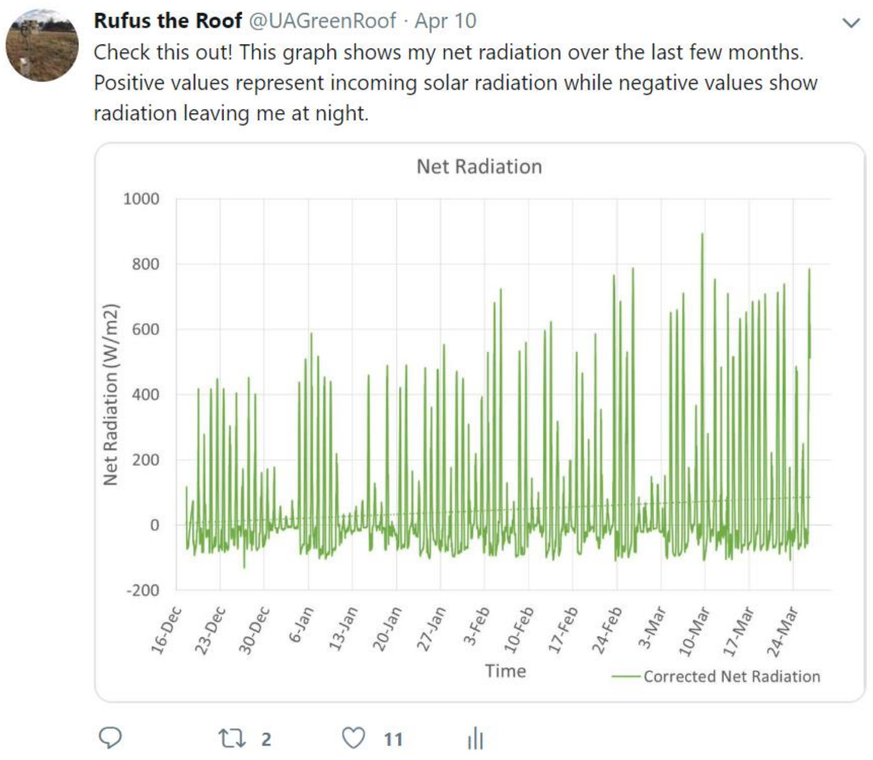

Rufus the Roof @UAGreenRoof · Mar 28
I've been quietly collecting loads of data! Here's a graph showing my 4 inch soil moisture profile with corresponding rain events over the last 4 months!

Image 8: Volumetric water content data tweet



Rufus the Roof @UAGreenRoof · Apr 10

Check this out! This graph shows my net radiation over the last few months. Positive values represent incoming solar radiation while negative values show radiation leaving me at night.

Image 9: Net radiation data tweet

3. Twitter Analytics

Twitter has equipped users with a tool that allows the tracking of the reach and engagement levels of an account called “Twitter Analytics.” The @UAGreenRoof account was created in January, 2019. From the date of creation to April, 21, 2019 it has produced 8 tweets and achieved the following statistics relating to overall reach and engagement:

- 116 followers
- 14,008 impressions (number of times users encountered tweets from the account)
- 769 profile visits

The amount of profile visits is a key indicator of the success of the design. As this value increases the overall impact of the project increases as well. It shows that members of the public engage with scientific information regarding LID infrastructure that they might not have previously. As engagement increases, so too does the intrinsic educational power of a live data tweeting system.

V. DISCUSSION AND FUTURE OPPORTUNITIES

The project is far from being truly complete. The initial goal of full automation was left unrealized. However, the structure for its realization is in place. There are more alternatives to be decided regarding how to achieve automation. They appear to be realistic endeavors especially considering that consistent connections between the datalogger, the Raspberry Pi, and the University Wi-Fi have been established.

1. Automation of Captured Rainfall Volume Tweeting:

The randomized Python script only requires one input in order to operate. This input is the cumulative rainfall depth from the previous day. A value could be automatically pulled from the NOAA – National Climatic Data Center's (NCDC) data system and imported into the Python script via the NCDC Web Services API on a daily basis. The Python scripts for captured rainfall volume tweeting have an imbedded "else" function that prevents the script from publishing a tweet if no precipitation occurred the previous day. Under manual operation this function is inept because rainfall depth is manually inputted the day following a known precipitation event. If automated fetching from the NCDC server were to be established this function would be essential.

There are many developed software that can be used to schedule the running of Python scripts. On the Raspberry Pi the most common method for automation is the "crontab" program that is built in to the operating system. This program is able to schedule commands in the Raspberry Pi command terminal. So with crontab it would be possible to schedule a command that fetches the rainfall depth, imports it into the captured volume Python script, and publishes the volume in a tweet.

Drake Field is several miles from the Hillside Auditorium green roof. Because of the geographic distance there is inaccuracy in using rainfall depths captured there as the assumed depths captured at the green roof. One potential solution to the inaccuracy would be to add an electronic rain gauge to the Raspberry Pi or CR1000 that feeds on site rainfall totals into the Python script via crontab automation. A rain gauge is the ideal alternative for the accuracy of the sensing station. However, the decision of whether to use it or the NCDC Web Services API would depend on rain gauge cost.

2. Potential Future Applications:

The Raspberry Pi microcomputer has shown itself capable of serving as a remote file transfer device for scientific dataloggers. However, it is capable of far more. There are a myriad of sensors available that are made specifically for using the Raspberry Pi as a datalogger itself. These cheap sensors could be calibrated with the more accurate ones produced by Campbell Scientific and deployed on a wider scale. The Pi has its processing power limits, but when power demands are low, it can serve as a cost effective measurement and control device. As mentioned in the literature review, some colleges are looking to the Raspberry Pi as a central element in the design of smart campuses. If these were deployed with sensor arrays across the campus they could communicate with one another and publish tweets to inform students of all manners of things regarding the campus, from study room vacancy to overall foot traffic. These technologies could be designed and deployed at a remarkably low cost to universities. The concept of a smart campus could also be extended to industry applications. Widespread data collection in industrial settings could deliver real time information relating to process inefficiencies.

The low cost and capabilities of this technology could also be utilized beyond an entertaining, educational setting. Cities could have a vested interest in installing systems like the one developed in this thesis for the organization and deployment of maintenance operations. Green roofs and other LID structures have imbedded maintenance costs associated with issues such as drain clogging or regular landscaping. Sensors could be used to actuate maintenance with simple messages relating to the sensed problems. Maintenance crews likely have little use for data and graphs, but if a quick blurb detailing what the LID structure needs could simplify a city's maintenance dispatch network. This would reduce overall costs associated with damage that goes unnoticed and inefficiencies in maintenance dispatching.

There are potential issues associated with the widespread deployment of sensors. In the agriculture industry some companies are developing proprietary sensing networks that deliver real time information to farmers. This is good for farmers and allows them to better understand their production. However, it also opens the doors to the sensors suppliers collecting metadata and developing algorithms to know what a farm's yield would be before the farmer knows. The sensor supplier could then use this data to hedge bets against them in the open market. I believe that in order to maintain the beneficial functionality of these sensing networks there must be a disconnect from massive investors that have stakes in the sources of the data and could negatively impact those that use their products. Open source technology like the Raspberry Pi could solve this problem. The technology could be deployed in a manner that makes setup streamlined while maintaining distance from large agricultural corporations.

For now, Rufus the Roof will remain steadfast on the Hillside Auditorium green roof, weathering storms and informing the public on the benefits of LID design. There may be a future when Rufus no longer needs to be operated manually, when data flows in automatically and smoothly. There may also be a future where Rufus is not alone, and other research stations on campus join the fold. That realization will require future students with interest in the automated conveyance of data. It is the hope of this thesis that Rufus could serve as an inspiration of that interest just as the tweeting tree in Gent inspired this project. There is something special about inanimate objects developing personalities and the ability to communicate scientific data in an entertaining way. Rufus the Roof might be the first tweeting LID on the University of Arkansas campus, but assuredly will not be the last.

VI. ACKNOWLEDGEMENTS

I would like to thank Dr. Benjamin R. Runkle for his mentorship and advising throughout the project.

I would also like to thank Dr. Thomas A. Costello for loaning the Raspberry Pi and accompanying equipment as well as joining my defense committee, Colby Reavis for assistance in connecting the Raspberry Pi to the CR1000 when the project was beginning to seem impossible, Dr. Kosana Suvočarev for early assistance with station setup and CR1000 programming, and the University of Arkansas Honor's College for funding the overall research project.

The idea of this project would not have existed without a study abroad experience in Gent, Belgium during the summer following my freshman year. Without Honors College Fellowship funding and assistance from Fellowship coordinator, Kelly Carter, I would not have gone to Gent and would not have encountered that tweeting tree. That trip to Gent also allowed me to meet and become great friends with Dr. Curt Rom. His friendship and guidance has helped to propel me through college. I would like to thank him for joining my committee and for showing me the wonderful city of Gent and its strange, sentient tree.

VII. REFERENCES

Bowman, T., & Thompson, J. (2009). Barriers to implementation of low-impact and conservation subdivision design: Developer perceptions and resident demand. Landscape and Urban Planning, 92(2), 96-105. doi:10.1016/j.landurbplan.2009.03.002

Hentschel, K., Jacob, D., Singer, J., & Chalmers, M. (2016). Supersensors: Raspberry pi devices for smart campus infrastructure. Paper presented at the 58-62. doi:10.1109/FiCloud.2016.16

Mercer, K. L. (2018). Citizen science. American Water Works Association. Journal, 110(6), 2-2. doi:10.1002/awwa.1093

Miguez, M. G., Rezende, O. M., & Veról, A. P. (2015). City growth and urban drainage alternatives: Sustainability challenge. Journal of Urban Planning and Development, 141(3), 4014026. doi:10.1061/(ASCE)UP.1943-5444.0000219

Neumann, M., Mues, V., Moreno, A., Hasenauer, H., & Seidl, R. (2017). Climate variability drives recent tree mortality in europe. Global Change Biology, 23(11), 4788-4797. doi:10.1111/gcb.13724

Prince George’s County (PGCo). (1999). Low-Impact Development Hydrologic Analysis. Department of Environmental Resources, Prince George’s County, Maryland.

Steppe, K., von der Crone, J., & Pauw, D. (2016). TreeWatch.net: A water and carbon monitoring and modeling network to assess instant tree hydraulics and carbon status. Frontiers in Plant Science, 7, 993. doi:10.3389/fpls.2016.00993

Westra, S., Fowler, H. J., Evans, J. P., Alexander, L. V., Berg, P., Johnson, F., Kendon, E. J., Lenderink, G., Roberts, N. M. (2014). Future changes to the intensity and frequency of short-duration extreme rainfall. Reviews of Geophysics, 52(3), 522-555. doi:10.1002/2014RG000464