Spring 5-18-2018

Remote Monitoring of Cherry Wetness Using a Leaf Wetness Sensor and a Wireless Sensor Network

Shyla Clark
*University of New Orleans, sclark6@uno.edu*

Follow this and additional works at: [https://scholarworks.uno.edu/td](https://scholarworks.uno.edu/td)

Part of the [Digital Communications and Networking Commons](https://scholarworks.uno.edu/academic/)

Recommended Citation


[https://scholarworks.uno.edu/td/2448](https://scholarworks.uno.edu/td/2448)

This Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UNO. It has been accepted for inclusion in University of New Orleans Theses and Dissertations by an authorized administrator of ScholarWorks@UNO. The author is solely responsible for ensuring compliance with copyright. For more information, please contact [scholarworks@uno.edu](mailto:scholarworks@uno.edu).
Remote Monitoring of Cherry Wetness Using a Leaf Wetness Sensor and a Wireless Sensor Network

A Thesis

Submitted to the Graduate Faculty of the University of New Orleans in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

by

Shyla Clark

B.A. University of Puget Sound, 2007

May, 2018
Acknowledgements

Dr. Christopher Summa, University of New Orleans
Dr. Adlai DePano, University of New Orleans
Dr. Stephen Ware, University of New Orleans
Dr. Minhaz Zibran, University of New Orleans
West Mathison, Stemilt Growers
Luke Schritter, Stemilt Growers
Mark Peaslee, Stemilt Growers

…
Contents

List of Figures ........................................ vi
List of Tables .......................................... viii
List of Abbreviations ................................... ix
Abstract .................................................. x

1 Introduction ........................................... 1
  1.1 Problem Definition .................................. 1
      1.1.1 Project Motivation .............................. 2
      1.1.2 Problem Statement .............................. 4
  1.2 Proposed Solution .................................. 5
      1.2.1 Research Objective .............................. 5
      1.2.2 Project Scope ................................... 6
      1.2.3 Project Contribution ............................ 6

2 Background ........................................... 7
  2.1 Industry Analysis .................................. 7
      2.1.1 Wireless Sensor Networks in Orchards .......... 7
      Uses ................................................ 9
      Materials .......................................... 10
2.1.2 Leaf Wetness Sensors in Orchards .............................................. 10

2.2 Literature Review ........................................................................ 12
  2.2.1 Wireless Sensor Networks in Precision Agriculture ................ 12
  2.2.2 Leaf Wetness Sensors in Orchards .......................................... 12

Use Cases ....................................................................................... 13

3 Materials and Methods ................................................................. 14
  3.1 Communication ......................................................................... 14
    3.1.1 Protocol .............................................................................. 14
    3.1.2 Radios and Antennas .......................................................... 15
  3.2 Hardware .................................................................................. 15
    3.2.1 Base Station ........................................................................ 15
      Coordinator Radio ...................................................................... 17
      Computer .................................................................................. 18
      USB-to-Serial Converter ........................................................... 18
      Power Supply ........................................................................... 19
    3.2.2 Wireless Sensor Node .......................................................... 19
      Router Radio .............................................................................. 20
      Microcontroller ......................................................................... 20
      Shield ....................................................................................... 20
      Dialectic Leaf Wetness Sensor ................................................... 22
      Power Supply ........................................................................... 22
  3.3 Software and Firmware ............................................................. 23
    3.3.1 Wireless Sensor Node .......................................................... 23
      XBee Radio Configuration ......................................................... 23
      Arduino Program for Reading Sensor ......................................... 24
    3.3.2 Base Station ........................................................................ 24
3.4 Testing Procedure ................................................................. 26
  3.4.1 Environment ................................................................. 26
    Atmosphere ................................................................. 26
    Installation ............................................................... 26
  3.4.2 Test Inputs ................................................................. 27
    Simulated Bird Droppings ............................................. 27
    Simulated Rain .......................................................... 27
  3.4.3 Intervals ................................................................. 27
  3.4.4 Tests ................................................................. 27

4 Results and Discussion .................................................. 31
  4.1 Results ................................................................. 31
    4.1.1 Clean Sensor Results ........................................... 31
      Control ............................................................... 31
      Drops ............................................................... 31
      Sprays .............................................................. 33
    4.1.2 Dirty Sensor Results ........................................... 34
      Control ............................................................... 34
      Drops ............................................................... 36
  4.2 Discussion ............................................................. 37
    4.2.1 Clean Sensor Results ........................................... 37
      Control ............................................................... 37
      Drops ............................................................... 37
    4.2.2 Sprays ............................................................ 37
    4.2.3 Dirty Sensor Results ........................................... 38
      Control ............................................................... 38
      Drops ............................................................... 38
5 Conclusion and Future Work

5.1 Conclusion .................................................. 39
5.1.1 Limitations ............................................. 39

5.2 Future Work ................................................ 39
5.2.1 Network Hardware .................................. 40
5.2.2 Network Communications ......................... 41
5.2.3 Leaf Wetness Sensor ................................. 41

Bibliography ................................................. 42

Vita ......................................................... 45
List of Figures

1.1 Sweet cherry cracking .............................................. 3
2.1 Wireless Sensor Network in an Orchard .......................... 8
2.2 AgWeatherNet station locations with temperatures .......... 11
3.1 Raspberry Pi with XBee Explorer and XBee .................. 16
3.2 XBee S2C Radio Module ........................................... 17
3.3 Sparkfun XBee Explorer USB-to-Serial ......................... 19
3.4 Wireless Sensor Node ............................................. 20
3.5 Arduino Uno Microcontroller Board ............................ 21
3.6 SparkFun XBee Shield for Arduino ............................. 22
3.7 Decagon Dialectic Leaf Wetness Sensor ......................... 23
3.8 API Frame .......................................................... 24
3.9 XCTU in Network Working Mode for visualizing network .... 25
3.10 Water Application Test: Sprays ................................. 28
3.11 Water Application Test: Drops .................................. 29
3.12 Water Application Test: Dirty Sensor ......................... 30
4.1 Clean Control ....................................................... 32
4.2 Clean 3 Drops ..................................................... 33
4.3 Clean 3 Sprays ...................................................... 34
4.4 Dirty Control ....................................................... 35
List of Tables

3.1 Arduino Uno Rev3 Specifications ..................................................... 21
3.2 API Frame Bytes ............................................................................. 26
3.3 Water Application Tests: Drops ......................................................... 27
3.4 Water Application Tests: Dirty Sensor ............................................... 28
3.5 Water Application Tests: Sprays ......................................................... 29
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>LWS</td>
<td>Leaf Wetness Sensor</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RSSI</td>
<td>Relative Signal Strength Indicator</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
Abstract

To get the best prices, sweet cherry growers must supply blemish-free fruit. Unfortunately, mature cherries have a fragile composition, rendering them susceptible to damage from heat, wind, birds, and rain. Rain is particularly devastating, because cherries split when they absorb too much water. Since rainstorms are common in the otherwise arid regions where most cherries are grown, growers must have a system for quickly deploying rain removal methods. The current industry solution relies on human observation and implementation, which is error-prone and costly. This project proposes an automated cherry wetness system using a Decagon Devices leaf wetness sensor (LWS) and a wireless sensor network (WSN). The research consists of analyzing industry and literature for uses of WSNs and LWSs in orchards and testing a LWS in a prototype WSN. The system will be evaluated for its potential to provide a precision-agriculture solution to the problem of remote cherry wetness detection.

Keywords: Washington state, sweet cherry, cherry split, leaf wetness sensor, wireless sensor network, ZigBee, Arduino, precision agriculture, orchard automation
1 Introduction

1.1 Problem Definition

The United States produces nearly 900 million pounds of sweet cherries per year [1], making it the second-largest sweet cherry producer in the world [1], and the state of Washington is responsible for nearly 60 percent of that [2]. The Washington state sweet cherry growing season is short with only 60 days between bloom and harvest, and because cherries ripen on the tree, they are evaluated daily until they meet the requirements for sugar-content, color and size. Once they are deemed perfectly ripe, they are picked, packed and shipped. Cherries are highly perishable, so their journey from farm to market happens in a matter of days – even if destined to an overseas market. During this relatively short harvest window, prices fluctuate dramatically due to spikes in supply that can happen overnight. It takes 50,000 people roughly 90 days to pick a single harvest, and growers, packing houses and distributors must each carefully plan and strategize how to handle the deluge. The product changes hands several times before reaching grocery stores, and at no point along the way are cherries able to be stored. It is imperative that the industry automate and streamline as many systems and processes as possible in order to get sweet cherries to consumers in excellent condition.
1.1.1 Project Motivation

The idea for this project came from a specific problem in the sweet cherry industry. Sweet cherries grown for the fresh market are like table grapes in that the appearance of the fruit matters as much as the taste. Unfortunately, cherries become more fragile as they mature. This adds another layer of complexity to an already complex set of requirements. Orchardists growing sweet cherries must care for not only the thousands of trees in their orchard but also the millions of individual fruits as well. Due to the challenges inherent to growing cherries, a good crop of cherries can be very lucrative; on the other hand, cherry crops are easily ruined, especially in the third stage of growth [3]. Mature cherries on the tree are at risk for overheating, bird damage and, worst of all, skin splits from water absorption.

Despite months of careful tending, including expensive soil inputs, pest control systems and irrigation management, a single rainstorm late in the growing season can render an otherwise perfect crop completely valueless. Specifically, water can ruin a mature cherry when it is allowed to pool in the bowl near the stem for too long. When a cherry absorbs too much water too fast, the flesh will swell and the skin will inevitably split [3]. These splits can occur on the tree, on the way to the packing house, or even after they’ve been packed and shipped overseas [3]. Because a box of cherries will change hands several times before reaching its final destination, more than one party in the supply chain has an interest in making sure splits do not occur. Unfortunately the farmer bears the most responsibility in preventing splits, because the primary cause of splitting is moisture that remains on the fruit for an extended period of time [3]. Due to the cherry’s natural shape, with a bowl at the top and a point at the bottom, when rain falls, it will collect in those two areas. When this happens, the cherry will absorb it and eventually split.

In Eastern Washington where most of the world’s sweet cherries come from, growers use two rain removal techniques. The cheaper method is a tractor attachment that blows air up and into the orchard canopy. This can be done relatively safely at night and in the rain. Helicopters provide
a more efficient drying method in which the aircraft slowly traverses the orchard, staying low enough that the wind generated by the propeller can blow rainwater off the orchard canopy from above. Helicopters are the quickest and most successful cherry drying method available, and some of the larger growers have their own fleet of helicopters dedicated to drying cherries [4]. A single helicopter and pilot on standby can cost thousands of dollars per day, so smaller growers who don’t have as many acres to protect typically rely on a helicopter service that must be ordered as needed, not unlike a city cab.

Irrespective of the actual drying method, one of the most critical components of any rain management program is recognizing that rain is affecting cherries in the first place. The current method of rain recognition is neither automated nor data driven – in fact, it’s usually a human who must be alert both day and night. The orchard manager typically keeps track of the weather using AgWeatherNet, a website developed by Washington State University [5]. AgWeatherNet has weather stations scattered throughout most of the Eastern Washington’s top growing areas.
Eastern Washington’s geography is diverse, and these weather stations provide more sophisticated weather data than radio stations. Even so, the weather data provided is not site-specific, so when overnight weather events are predicted, orchard managers must sleep near high-risk cherry blocks. They must wake up periodically to check for rainfall – sometimes as often as every two hours. If rain is detected, the manager must go into the orchard to evaluate how wet the cherries are actually getting. Then he or she will decide whether or not tractor drivers need to be deployed to start blowing the orchard. This is also the point at which someone must alert the helicopter company that services may be necessary. For smaller farms that cannot afford to have a dedicated helicopter team on-site, calling the helicopter company is an important step. As farmers call in to the company, the dispatcher adds each to the queue. Water must be removed as soon as possible, but the helicopters don’t dispatch at night or while it’s still raining; therefore it is essential to call in as soon as rain is detected in order to get a higher priority in the queue.

1.1.2 Problem Statement

Cherry harvest is a hectic grind with many layers of complexity. Cherry orchard managers make important decisions during the daily harvest requiring them to be rested and alert, yet they must also be alert for overnight rainfall. They are responsible for noticing the onset and duration of rainstorms as well as for alerting and deploying of rain management strategies. These tasks are logistically difficult, especially when dealing with orchards scattered across several miles. The overnight vigils are physiologically taxing. It is also costly for farm owners to pay managers overtime, particularly because managers are often some of the highest-paid workers in the company.

1.2 Proposed Solution

Orchard managers must deal with several harvest-related systems – all with high-levels of complexity. A location-based rain alert system would enable them to get more sleep overnight if they
didn’t have to wake up every few hours to check for weather events. It would also solve the problem of detecting rainfall in remote orchards. Relieving these managers would also lower operating costs by reducing their overtime pay. The complete system, first and foremost, should alert users immediately at the onset of rain and display the intensity. It should also record the duration and amount of rain for that particular weather event. It should also alert users when the rain stops.

The proposed solution is to deploy a remote node with a rain detection sensor in a vulnerable part of the orchard. Using radio signals the node would communicate with another base station node that is connected to internet. The base station would receive the data stream and generate alerts which could then be sent to a user. Finding a sensor that can accurately convey this information without false-positives is as important as deploying the sensors in a reliable wireless network.

This solution is different from other weather alert applications on the market, because the data is site-specific and doesn’t rely on third-party weather stations [5].

1.2.1 Research Objective

The objective of this project is to test Decagon’s leaf wetness sensor (LWS) for suitability as part of a cherry wetness alert system through running it in a prototype wireless sensor network (WSN). Rain gauges have been on the market for decades; however, most of them haven’t been reliable enough to warrant building an entire system around them. Decagon’s Phytos 31 LWS provided a potential solution.

1.2.2 Project Scope

The scope of this project is limited to testing a wireless sensor network and a sensor in a lab environment.
1.2.3 Project Contribution

The contribution of this thesis project is to test an industrial-grade, orchard leaf wetness sensor in a novel application to solve the long-standing problem of monitoring pre-harvest cherry wetness.
2 Background

2.1 Industry Analysis

Stemilt Growers, the largest fresh market sweet cherry shipper in the world [6], [7], served as an case study representing the current industry practices. The best practices of this large company were compared against a paper describing the practices of a well-known competitor, Auvil Fruit Company. Stemilt is an industry titan, and Auvil Fruit Company is well-known for being an early-adopter of innovative growing practices [8].

2.1.1 Wireless Sensor Networks in Orchards

Orchards are divided into blocks, and a single orchard might consist of one block or hundreds. There is no defined system for how to divide an orchard into blocks, because, like cities, farms change over time. The general standard is that a block contains a planting of trees with the same specifications. The more uniform the trees are, the easier they are to manage. Every block will contain trees planted at the same time, on the same rootstock and of the same variety. Plantings are made in uniform rows to allow ease-of-access when the trees reach maturity. Large commercial growers might have a huge orchard divided into 10-acre blocks, whereas a smaller grower could divide their orchard into 2-acre blocks. How orchards are divided is a matter of personal preference, but it is generally performed with management in mind.
The existing block system guides the design of WSNs in orchards. Managers choose a sensor node installation site to represent a section of a block, a block or the entire orchard. Depending on differences in soil and terrain, more than one sensor node might be necessary for a single block. For example, in the Columbia Basin area, where land is flat and the soil relatively unvarying, a single well-placed node might be able to represent a uniform 10 acres, whereas in Cascade Foothills area, where the terrain varies and blocks might be tucked into canyons and along north or south-facing slopes, a manager might need to deploy multiple sensors of the same type in order to get an accurate representation of a block.

For example, trees planted at the top of a hill will have different irrigation water requirements than trees planted at the bottom. The uppermost layer of soil is always wet after an irrigation set, so managers like to use soil moisture sensors placed at the top and bottom of a hill to determine
how much water is reaching the roots and how much is traveling downstream.

Uses

Stemilt uses wireless sensor networks primarily for collecting data about irrigation. The company is most concerned with soil moisture at multiple depths as it relates to the amount and frequency of running irrigation. Variances in soil types and plantings require customized irrigation programs. The orchard blocks are spread across 7,000 acres of diverse geography, so very few orchard blocks have similar specifications. Managers aim to tailor their irrigation program to each block’s individual needs, and eventually they develop an intuition based on watering schedule versus rates of growth. For the large orchards, this can be tricky. One manager might rely on several subordinates who are responsible for turning irrigation valves on and off. It’s easy to accidentally miss a valve, and without any reliable data stream coming in, it’s almost impossible for anyone to catch such mistakes quickly. With a soil moisture probe, strange spikes or dips in the data stream indicate watering patterns. Stemilt combined the soil moisture probe with pressure sensors in the irrigation lines to enable them to see the correlation between running irrigation sets and the amount of soil moisture at four-inch intervals from four to 24 inches deep.

Auvil Fruit Company also uses soil moisture sensors in two configurations: one is their proprietary configuration using Neutron Probe with unknown network architecture, and the second is a test system consisting of a Decagon Devices Em50G datalogger and Decagon soil moisture sensors[8]. The Decagon device relies on cellular data telemetry, and independently transmits the data stream from its sensors to its remote web server[9]. These devices are not networked with each other and act as their own base station[8].
Materials

The original wireless sensor networks used at Stemilt were set up by a consultant firm specializing in orchard technology. When Stemilt decided to expand their coverage in 2017, they continued to use the original system architecture. The system they use consists of Banner DXM100 radios, multidirectional antennas, ModBus protocol, 900 mHz ISM frequency band, and a combination of solar and battery energy supplies. The software Stemilt uses to disseminate this data to managers is OnFarm[10], which provides a web application with customizable widgets for viewing data streams in realtime.

2.1.2 Leaf Wetness Sensors in Orchards

Stemilt does not use LWSs in any of their deployed networks, and it is unknown whether or not Auvil Fruit Company does. Managers at both companies use data from AgWeatherNet weather stations.

AgWeatherNet weather stations are managed by Washington State University Irrigated Agriculture Research and Extension Center in Prosser, Washington. AgWeatherNet has 176 stations scattered across Washington state with the most density in the arid regions of the Columbia River Basin where most of the irrigated agriculture takes place[5]. These weather stations are comprised of CR-1000 data loggers and approximately ten sensors including Model 014A Met One Wind Speed Sensor, Model 024A Met One Wind Direction Sensor, Rain Gauge Tipping Bucket TR-525I Rainfall Sensor and 237 Leaf Wetness Sensor – all from Campbell Scientific[5]. Instrument readings are taken every five seconds and the data is aggregated every 15 minutes and sent to a web server via cellular data telemetry[5].

For most growers, AgWeatherNet provides much-needed access to remote location-based weather data, because such sophisticated monitoring stations would be too costly to implement and manage in-house. These weather stations are miles apart and cannot be used to represent weather
Figure 2.2: AgWeatherNet station locations with temperatures[5]
events within specific blocks[11]. Perhaps the greatest benefit of these weather stations are the collections of historical data and the data models that have been trained using them. AgWeatherNet uses the data collected by their own LWS from Campbell Scientific to train models for rain and wetness-related diseases like Fire Blight[5]. These models provide useful parameters for growers about disease prevention, which, if they had their own leaf wetness sensors, would be even more helpful. AgWeatherNet has created a variety of models based on sensor data which are referred to as decision aids for growers[5].

Like most of the other LWSs on the market, Campbell Scientific’s sensor must be calibrated and painted before use[12].

2.2 Literature Review

A survey of academic research was conducted to get an understanding of how wireless sensor networks and leaf wetness sensors are already being tested and used in precision agriculture applications.

2.2.1 Wireless Sensor Networks in Precision Agriculture

The most common WSN communication protocol is ZigBee[13], [14], [15], [16].

2.2.2 Leaf Wetness Sensors in Orchards

LWSs have been tested in orchards since the 1970s for automation applications[17], but none have been used to detect cherry wetness.
Use Cases

The primary and most common use case for WSNs in orchard environments is to monitor irrigation strategies and their affect on the moisture content of the soil[13], [14], [16]. The second most common use is to monitor pressure in the irrigation lines to determine when irrigation is running in a particular block and for how long. The third most common use case is to monitor pumps that facilitate the irrigation in order to receive alerts if the pump malfunctions.
3 Materials and Methods

3.1 Communication

3.1.1 Protocol

The first step in constructing the WSN was to select the protocol. *ZigBee protocol* was chosen because it allows for wireless mesh networking as well as API packet formation.

Mesh networking is essential in orchard networking applications, because remote nodes should not depend on a single route to the base station. The WSN must be self-healing, meaning it must have multiple communication routes should one or more routers go down. It is most cost-effective to have sensors attached to every node deployed, and interruptions in data streams are a sufficient means for recognizing a down node; however, this strategy depends on all working nodes to be able to transmit their data to the base station successfully.

With multiple data streams transmitting in a mesh network, the base station must be able to parse which node in the network sent the original message regardless of how many nodes the message routes through. The base station must also be certain that the message it receives is complete and not API packets provide this functionality.
3.1.2 Radios and Antennas

For the scope and budget of this study, neither the radios nor the antennas needed to be long-range or professional grade, so a pair of Digi XBee S2C radios were chosen. These radios ship with permanently affixed wire antennas, which are sufficient for indoor prototyping. In orchard applications, the antenna is typically affixed high above the tree canopy with a cable connecting it to the radio which is installed at eye-level for ease-of-use.

3.2 Hardware

Materials were selected on the basis that they needed to have the potential to be used in the field, although they did need to be rated as such. Cheaper approximations of industrial grade hardware were used.

3.2.1 Base Station

The base station is the cornerstone of any WSN that requires cloud storage for data streams or a web-based interface. The base station acts as the edge device, obscuring the other nodes in the network from the internet. In this sense, it functions as a firewall and a router. The base station is also sometimes referred to as the gateway, hub, or field gateway. In field applications, it might have a radio for communicating with the other nodes in the network as well as an internet connection via a cellular card or WiFi. The base station used in this experiment consisted of a coordinator radio, computer, USB-to-serial converter, and power supply.
Figure 3.1: Raspberry Pi with XBee Explorer and XBee
Coordinator Radio

In this instance, a *XBee ZigBee S2C* radio was used and configured as the network coordinator. The XBee ZigBee S2C is a transceiver, which means it has transmitter and receiver circuitry in the same housing. Every network requires a single coordinator radio. The coordinator is responsible for starting the network, selecting the channel and setting the Personal Area Network ID (PAN ID). It assigns all of the radios in its network a network address. It cannot sleep and must be powered on at all times. In any network with multiple radios sending data to the coordinator, the coordinator must be in API mode to receive API packets which contain the sender’s address. A matching pair of radios were chosen for this experiment, but radios from other vendors would also work provided they communicate with the same ZigBee protocol.

![XBee S2C Radio Module](image)

- Outdoor line-of-site radio frequency (RF) range: 4,000 ft
- Operating frequency band: Industrial, Scientific and Medical (ISM) 2.4 - 2.5 GHz radio band
- Transceiver: Silicon Labs EM357
• Interface options: UART 1 Mb/s maximum (burst) and SPI 5 Mb/s maximum (burst)

• Operating temperature: -40 to 85 degrees C (industrial)

There is an important distinction between XBee S2 and S2C radios: Series 2 (S2), product family XB24-ZB, is an older module that frequently appears in tutorials. These used a smaller processor that required six different firmware versions to offer Coordinator, Router and End devices code in either AT or API mode. The new hardware in S2C uses a larger processor that has all six versions in one[18]. That is if you want Router AT then you do nothing at all (default). If you want API mode, you set the API command to 1 or 2. If you want a Coordinator you set the CE command to 1 (0 is router or end device). For End device, set the SM command to a value greater than 0. All of this is covered in the current product manual for the XBee ZB S2C hardware.

Computer

The base station radio is capable of coordinating and programming remote nodes. It also receives the data streams and acts as an internet gateway. To get the most out of a WSN, the base station should have an interface that allows users to easily manage their network. A Raspberry Pi 3 was trialed during this project, but ultimately a MacBook Pro was used to conduct the tests. The Pi required three peripherals – a mouse, a keyboard, and a monitor. This rendered it impractical for the scope of this project. The Raspberry Pi 3 had WiFi and presents an exciting option for future work.

USB-to-Serial Converter

To connect the Coordinator radio to the computer acting as the base station, a SparkFun XBee Explorer USB was used. This dongle is appropriate regardless of the computer selected and was also used to program the XBee.
Power Supply

The XBee was powered by the MacBook Pro via the XBee Explorer USB.

3.2.2 Wireless Sensor Node

A wireless sensor node is a self-contained, deployable unit sometimes referred to as a "mote" [19]. A single network might contain hundreds of motes, each capturing sensor data or controlling actuators in remote locations and communicating wirelessly with the network gateway. Because this study was only interested in the behavior of a single sensor, a single mote was used. It is important to point out that sensors and actuators may be connected to a base station; however, a remote base station would require internet connectivity, which may be neither cheap nor feasible in certain installation locations.
In this instance, a XBe... configuration and extension options[20]. Arduinos are not typically used in industrial applications, because they are considered more fragile than microcontrollers developed specifically for industrial use[21].

**Shield**

A SparkFun XBee Shield for Arduino Uno was used to interface with the Arduino Uno. The shield does not ship with headers, so headers had to be soldered on before it could interface with the Arduino board. Using a pre-made shield was advantageous. The Arduino has only a single serial port, but the shield has a built-in serial select switch. This switch allows you to the change
the XBee’s serial pin (DIN, DOUT) connections between DLINE: Software serial port to upload sketch from computer to Arduino via USB and UART: Hardware serial port for communicating with other XBees[22]. It also has the following useful LED indicator lights[22]:

- DIO5: blinks green when paired with another XBee
- DIN: indicates wireless data being transmitted
- DOUT: indicates wireless data being received (red)
• RSSI: indicates relative signal strength of last received transmission and brighter when XBees are closer together

Dialectic Leaf Wetness Sensor

A dialectic leaf wetness sensor made by Decagon Devices out of Washington state was chosen.

Power Supply

For this experiment, the primary power source was USB plugged into a wall outlet.
3.3 Software and Firmware

3.3.1 Wireless Sensor Node

XBee Radio Configuration

To configure the radio, a *SparkFun XBee Explorer USB* was used to interface between the radio and a laptop. Digi, the maker of XBee radios, has a proprietary program called XCTU for configuring XBee radios and visualizing and testing networks[23]; however, using a GUI program to configure and test the radios isn’t necessary[18]. Both of the radios used in this project were initially configured using the commandline interface on a MacBook Pro and Microsoft Surface Pro.

- Product family: XB24C (not XB24-ZB from older XBee S2 module)
- Function set: ZIGBEE TH Reg (TH for "Through Hole" (rather than SMD) and Reg for Regular not ZigBee Pro)
- Default mode: AT Transparent mode
S2C modules can communicate with older modules if all running compatible firmware (i.e. DigiMesh, ZB, etc..).

Arduino Program for Reading Sensor

An Arduino program was written to read the sensor connected to analog pin 0 (A0) and send it to the coordinator radio. Because the XBeee radio was configured for Transparent Mode, it was sufficient to write a program that printed to the serial port, knowing that whatever was printed would be packetized and sent to the Coordinator radio as per the radio’s configurations.

3.3.2 Base Station

XCTU was the software used in this project to receive and translate the incoming packets. At the time of this writing, XCTU was not available for Raspberry Pi’s Debian operating system. XCTU was chosen for three reasons: The first reason was that it has a graphical user interface which provided convenient network visualization for troubleshooting network errors[18]. The second reason is that it could be used to parse and translate the incoming hexadecimal serial packets into ASCII[23]. The third reason was that you could use it to run AT Commands on the XBee radio to reconfigure it if necessary[18]. While the commandline interface was fairly simple to use for radio configuration, it was more convenient to have all three of these tools in a single piece of software that could run on both Mac and Windows.

---

<table>
<thead>
<tr>
<th>Byte</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16 to n</th>
<th>n + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>7E</td>
<td>00</td>
<td>1F</td>
<td>90</td>
<td>00</td>
<td>13</td>
<td>A2</td>
<td>00</td>
<td>41</td>
<td>88</td>
<td>8C</td>
<td>A7</td>
<td>00</td>
<td>53</td>
<td>01</td>
<td>4C</td>
<td>57</td>
</tr>
<tr>
<td>ASCII</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*LSW currently: 78*

---

**Figure 3.8: Sample ZigBee Receive Packet API Frame**
**Figure 3.9:** XCTU in Network Working Mode for visualizing network
TABLE 3.2: API Frame Bytes

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start delimiter</td>
</tr>
<tr>
<td>2,3</td>
<td>Packet length: total number of bytes except start delimiter, length and checksum</td>
</tr>
<tr>
<td>4</td>
<td>Frame type (e.g. Receive packet)</td>
</tr>
<tr>
<td>5 to 12</td>
<td>64-bit source address: MAC address</td>
</tr>
<tr>
<td>13,14</td>
<td>16-bit source address: network-assigned</td>
</tr>
<tr>
<td>15</td>
<td>Receive option</td>
</tr>
<tr>
<td>16 to n</td>
<td>Data</td>
</tr>
<tr>
<td>n + 1</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

3.4 Testing Procedure

3.4.1 Environment

The tests in this study were performed in a lab setting to completely control the sensor’s environment and establish a baseline for the sensor.

Atmosphere

The air temperature was 22.2 degrees Celsius with 34 percent humidity.

Installation

The tests were performed with the sensor installed on a level surface.
3.4.2 Test Inputs

Simulated Bird Droppings

To simulate bird droppings, Dijon mustard was warmed to 37.8 degrees Celsius and applied to a clean, dry sensor.

Simulated Rain

Distilled water was deemed superior to tap water for simulating rainwater due to its low mineral content[24]. The water was kept chilled to 3 degrees Celsius.

3.4.3 Intervals

The tests were performed for 500 seconds each with the base station receiving packets from the wireless sensor node.

3.4.4 Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clean sensor baseline</td>
</tr>
<tr>
<td>2</td>
<td>One 0.1 mL drop</td>
</tr>
<tr>
<td>3</td>
<td>One 0.1 mL drop and one 0.2 mL drop</td>
</tr>
<tr>
<td>4</td>
<td>One 0.1 mL drop, one 0.2 mL drop and one 0.3 mL drop</td>
</tr>
</tbody>
</table>
Table 3.4: Water Application Tests: Dirty Sensor

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Dirty sensor baseline</td>
</tr>
<tr>
<td>6</td>
<td>One 0.1 mL drop</td>
</tr>
<tr>
<td>7</td>
<td>Two 0.1 mL drops</td>
</tr>
<tr>
<td>8</td>
<td>Three 0.1 mL drops</td>
</tr>
</tbody>
</table>
Figure 3.11: Water Application Test: Drops

Table 3.5: Water Application Tests: Sprays

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Clean sensor baseline</td>
</tr>
<tr>
<td>6</td>
<td>One 2 mL sprays</td>
</tr>
<tr>
<td>7</td>
<td>Two 2 mL sprays</td>
</tr>
<tr>
<td>8</td>
<td>Three 2 mL sprays</td>
</tr>
</tbody>
</table>
Figure 3.12: Water Application Test: Dirty Sensor

(A) Dirty

(B) Dirty with water

Figure 3.12: Water Application Test: Dirty Sensor
4 Results and Discussion

4.1 Results

4.1.1 Clean Sensor Results

Control

A control was established on a clean and dry sensor with a surface temperature of 23.2 degrees Celsius.

No input– without contact from any outside materials, the value stayed between 70 and 71 for the full duration of the 500 second test interval. The value 70.5 was used as the value baseline.

Drops

The following were applied to a clean and dry sensor with a surface temperature of 23.2 degrees Celsius and a starting value of 70.

One Drops– when a single 0.1 mL drop was added, the value moved to 72 and 73.

Two Drops– when slightly larger 0.2 mL drop was added, the value moved to 75 where it stayed
Figure 4.1: Clean Control

Clean Sensor: Control (No Input)
until the third drop was added.

**Three Drops**– when the final and largest 0.3 mL drop was added, the value moved to 81 and 82.

![Clean Sensor: 3 Drops Distilled Water](image)

**FIGURE 4.2: Clean 3 Drops**

**Sprays**

The following were applied to a clean and dry sensor with a surface temperature of 23.2 degrees Celsius and a starting value of 71.

**One Spray**– when a single 2 mL spray was added the value moved to 93 and occasionally dipped to 92.

**Two Spray**– when another 2 mL spray was added, the value moved to 124 then to 122 and 121
where it stayed until the third spray was added.

**Three Sprays**—when the final 2 mL spray was added, the value moved to 143, 142 and 141 before dropping to 140 and 139 where it stayed.

![Graph](image)

**FIGURE 4.3: Clean 3 Sprays**

### 4.1.2 Dirty Sensor Results

**Control**

A control was established on a clean and dry sensor with a surface temperature of 23.2 degrees Celsius and a starting value of 70.
**Simulated Bird Droppings**— upon application of the matter, the value spiked to 176 then 226 and stayed between 227 and 228 for the duration of the interval.

---

**Figure 4.4: Dirty Control**

Drops

The following were applied around the matter on a sensor with a surface temperature of [xxx] degrees Celsius and a starting value of 229.

- **One Drops**— when a single 0.1 mL drop was added, the value moved to 231 and 232.
- **Two Drops**— when another 0.1 mL drop was added, the value moved to 234 where it stayed until
the third drop was added.

**Three Drops**—when the final 0.1 mL drop was added, the value moved to 235 and 236.

![Dirty Sensor: 3 Drops Distilled Water](image)

**Figure 4.5: Dirty 3 Drops**

### 4.2 Discussion

#### 4.2.1 Clean Sensor Results

**Control**

Without any external influences, the sensor did not deviate from its range of values: 70 and 71. Indicates that the value at 70.5 correlates with a dry sensor. Consequently, 70.5 was used as a baseline value.
Drops

What’s more promising is that the sensor is able to detect a single drop of moisture. For the first and smallest drop, the sensor value’s range only shifted by two. The same occurred with the second and third drops which moved the value by only 3 and 6 respectively. Had the drops been identical, the changes in value would most likely have been identical also.

4.2.2 Sprays

The sensor started with a value of 71 and the first spray of 2 mL of water increased the sensor value by 22. The second spray of 2 mL increased the sensor value by 30 and the third by only 18. The reason why these three applications of water might differ is due to the manner in which air is also forced out of the spray nozzle and may spray water off of the sensor. This does not make it an ideal method for testing the sensor, as most cherries on the tree are protected by thick bunches of leaves which would provide some manner of protection from forceful rainfall.

4.2.3 Dirty Sensor Results

Control

The sensor started with a value of 70 and after the matter was applied, the value had an initial spike of 106 then five seconds later increased by 50 then settled in at one and two higher than that. The huge spike in value after the matter was applied may have more to do with type of material applied than the volume. The matter was significantly more warm and denser than the distilled water from the prior experiments.
Drops

As in the drop experiments on the clean sensor, the dirty sensor also performed well, recognizing a single drop with a value shift of two. These drops were the same volume of water each, unlike in the clean sensor experiment, and the value changes reflected that with a two value increase after the second drop and a two value increase after the third drop.
5 Conclusion and Future Work

5.1 Conclusion

The results of this study prove that dialectic LWS can be used to measure cherry surface wetness.

5.1.1 Limitations

Although great care was taken to make the water applications as precise as possible, due to human error, there may be slight variances between the size of drops and between the size of sprays. There may also be slight variances in atmospheric, water and sensor temperatures, as the infrared temperature sensor used in this study is imprecise to two degrees.

5.2 Future Work

Future work with the hardware, network and software is necessary before the system can be used by industry. The system needs to be tested outdoors in an orchard setting under typical operating conditions. This will require further development of the prototype.

5.2.1 Network Hardware

Most of the hardware presented in this system is not robust enough for outdoor use.
The router nodes were developed for lab use only. As such, they require further development before they can be tested in outdoor environments. It is known that they will definitely require weatherproofing, enclosures, and different antennas.

Enclosures must be developed that can protect the Arduino from sprinklers and rain while providing user access. The node must be tested for moisture tolerance, and special actions may be necessary, like using a weatherproofing product like ToolDip to prevent moisture from reaching the electronics.

The microcontroller must also be tested for temperature tolerance. The ATmega328P microcontroller in the Arduino Uno R3 has absolute maximum ratings for operating temperatures of -55 degrees celsius to +125 degrees celsius\[25\]. The microcontroller needs to be tested at 50 degrees celsius ambient temperature, which is a typical Washington state high temperature in the summer.

The wire antennas used in this study are insufficient for industry applications, because they are fixed to the radio which will be placed inside an enclosure. The enclosures must be installed at roughly eye-level for ease-of-use, which places the enclosure box well below the top of the orchard canopy. Experience at Stemilt showed that antennas performed poorly when installed below the canopy and inside the waterproof enclosure; therefore, the nodes require larger multidirectional antennas that can be installed outside of the enclosure and above the canopy.

The LWS should be tested outdoors in an orchard environment so the sensor’s output after morning dew formation can be tested against rain outputs.

5.2.2 Network Communications

The network will require more testing and troubleshooting in a field environment. Testing the network for industrial applications requires additional routers, carefully chosen installation sites, and radio and protocol testing.
The base station XBee radio was configured in API mode to accept packets from multiple routers; however, only one router was tested in this prototype. To test the mesh networking capabilities of the network, more than one router must be used.

XBee radios with industrial grade antennas must be tested along with the ZigBee protocol to determine whether or not it will be robust enough for locations with changes in elevation. Additional routers may be necessary with this protocol, which may render the ZigBee choice less-desirable. To test the radios and protocol thoroughly, the nodes should be tested in a variety of installation site configurations.

5.2.3 Leaf Wetness Sensor

The leaf wetness sensor chosen for this study should be tested alongside different popular cherry varieties to create models.
Bibliography


Vita

Shyla Clark was born in Chelan, Washington. She earned her Bachelor’s degree in English and business administration from the University of Puget Sound in Tacoma, Washington. She joined the University of New Orleans computer science graduate program in 2015.