A novel system architecture for automated field-based tent systems for controlled-environment agriculture and experimentation

by

Dan W. Wagner

B.S., Kansas State University, 2017

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Co-Major Professor
Arslan Munir

Approved by:

Co-Major Professor
Mitchell Neilsen

Copyright

Abstract

Experimentation within the field of agronomy relies upon maintaining a controlled operating environment to determine various environmental factors' effects upon a crop. These experiments are carried out in small growth chambers and can control limited variables such as light, temperature, and humidity.Space is a premium inside the chambers which limits the capacity for additional sensors and other equipment. The chambers are set to specific environmental parameters and maintained throughout the experimental cycle. Field conditions are more complex than a growth chamber, which makes it difficult to analyze the effect of factors in a more realistic scenario. A system architecture for a field-based controlled environment for agriculture and experimentation is proposed. First, the overall architecture is proposed for integrating a multitude of wired and wireless sensors, different controllers, small unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs), and actuators to assess and maintain environmental variables. Second, each component is detailed for its role and responsibilities within the system. Next, a set of commercially available components are examined and compared for their strengths and weaknesses in the proposed system. Then, scientific applications of the system are proposed and explored.

We prototype and analyze a simplified implementation of the architecture in a wheat heat stress experiment, and demonstrate the capability of our proposed architecture in studying the effect of different environmental conditions on crops in CEA settings. The architecture is simplified to a set of two controllers connected over wireless local area network (LAN): one for heated/experimental tents, and one for the outdoor/control tents; three pairs of such controllers are incorporated into the experiment. A singular Raspberry Pi within each tent implemented the functionality of most of the components in the architecture via software modules. In the case study setup, each pair of controllers communicate with six temperature

sensors and one carbon dioxide sensor and continuously maintain the environment by managing relays to actuate a heater element by comparing the environment temperature with a pre-set temperature threshold.

Table of Contents

List of Figures

List of Tables

Nomenclature

I^2C Inter-Integrated Circuit

BEB Binary Exponential Backoff

BNN Binarized Neural Network

CEA Controlled Environment Agriculture

CPS Cyber-Physical System

GPIO General Purpose Input Output

HNT High Night-time Temperature

IOT Internet of Things

LAN Local Area Network

NDVI Normalized Difference Vegetation Index

RELU Rectified Linear Unit

TCP Transmission Control Protocol

UART Universal Asynchronous Receiver-Transmitter

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

VRNA Variable Rate Nitrogen Application

Chapter 1

Introduction

Experimentation within the field of agronomy relies upon maintaining a controlled operating environment to determine various environmental factors' effects upon a crop. These experiments are carried out in smaller growth chambers and can control variables such as light, temperature, and humidity. Space is a premium inside the chambers which limits the capacity for additional sensors and other equipment. The chambers are set to the specific requirements for the environment and not changed: they are maintained by the control system within the chamber itself. Field conditions are more complex than a growth chamber, which makes it difficult to analyze the effect of factors in a more realistic scenario. A larger, automated, field-based tent system designed for controlled experimentation proposed can solve these problems and make it easy for expansion.

Controlled Environment Agriculture (CEA) is an advanced form of agriculture based on hydroponics where crops grow inside of a controlled environment[1]. There is a constantly growing demand for fresh produce in the world that are pristine quality and have not been exposed to chemicals. Climate changes make growing desirable crops difficult when out of season, regardless of consumer demand. CEA remedies these problems by recreating the desired environment for the crop growth. Recent research developments in the field focus on optimizing horticultural lightning systems within greenhouses[2]. Cornell University's

GLASE consortium is continuing research into LED efficiencies and light wavelengths in greenhouses; their focus is creating integrated control systems for lighting, carbon dioxide, temperature, and humidity. More complex interactions, such as imaging data and aperture control, cannot be achieved easily without further research and revisions to existing systems. Higher-dimension data points, such as video/image data, are becoming more important and relevant in agricultural practices for increasing crop yield and quality through the use of drones and stationary cameras. A system architecture to manage the complex sensor systems and integrate them together is required for more informative experimentation.

Current academic research on CEA focuses on individualized, complex systems for analyzing single-dimension interactions between crops and the environment as well as larger-scale smart farming systems with multiple individual embedded devices scattered throughout the farm area. The former exhibits reliable environmental controls at the cost of space and complexity. Larger smart farming systems allow complex farming areas to be examined and regulated but are difficult to manage for environmental control decisions. Although recent years have seen growing progress in CEA[1], an architecture that can enable monitoring and control of multiple environmental factors in farm-like conditions still needs to be developed.

To overcome the limitations of small growth chambers, and to analyze the effect of different environmental factors on crops, we propose an architecture for large, automated, field-based tent systems for CEA and experimentation.The proposed architecture is modular and scalable comprising of different subsystems responsible for sensing, analytics, decision-making, and actuation. We discuss the constituent subsystems of the proposed architecture. A multitude of sensors, both wired and wireless, for observing various variables can be installed and used within the tent. The required environmental parameters for the area are designated by the designer or user of the system. After the parameters' specifications, the system uses the readings received from sensors to engage or disengage actuators to maintain an optimal environment.

The proposed system allows for the automated and controlled experimentation over a

larger area of crops. A multitude of sensors for various variables, both wired and wireless, can be installed and used within the tent. The required environmental parameters for the area are designated by the scientist to the system; once this is completed, the system uses the readings received from sensors to engage or disengage actuators to maintain the optimum environment. Actuators can be any device ranging from a robotic aperture used to move equipment to a propane heater for controlling temperature. Over time, the system learns from previous control decisions to enhance its decision-making capabilities and fine-tune the environmental parameters; this is achieved by using an artificial neural network. The neural network takes, as input, every reading from sensors connected to the system, as well as the actuator states. Based on these input parameters, it decides what needs to change to further optimize the area; the system learns about relationships between the parameters, which can be used for future research studies.

Chapter 2

Related Work

Recent research developments in CEA focus on optimizing horticultural lighting systems within greenhouses[2]. Cornell University's Greenhouse Lighting and Systems Engineering (GLASE) consortium is continuing research into light-emitting diode (LED) efficiencies and light wavelengths in greenhouses. Their focus is creating integrated control systems for lighting, carbon dioxide, temperature, and humidity. More complex interactions, such as imaging data and aperture control, cannot be achieved easily without further research and revisions to existing systems. Higher-dimension data points, such as video/image data, are becoming more important and relevant in agricultural practices for increasing crop yield and quality through the use of drones and stationary cameras. A system architecture to manage the complex sensor systems and integrate them together is required for more informative experimentation.

Current academic research focuses upon individualized, complex systems for monitoring singular variables or interactions between the crop and environment, as well as larger-scale Internet of things (IoT) based smart farming systems with multiple individual embedded devices scattered throughout the area. Castañeda-Miranda et al.[3] designed an IoT system to accurately predict frost levels in a greenhouse, and to activate an anti-frost irrigation system to prevent damage and crop disaster. The authors utilized an artificial neural network

(ANN) to predict the greenhouse temperature, which is then provided as input to a fuzzy logic system to determine the appropriate amount of anti-frost to apply to maintain the thawed environment. The fuzzy system incorporated the frost weather index and placed the ANN output into one of several bins based on the potential predicted hazard from the frost. By design, the system was able to power itself by using solar panels to charge the batteries connected to the embedded electronics. Their architecture[3] can leverage our proposed architecture to create a larger, overarching smart farm system that could monitor and control multiple environmental parameters by integrating the appropriate controller(s), actuators, and communication protocols. Farooq et al.[4] surveyed the role of IoT in agriculture for smart farming. They noted that agricultural trends were individualized systems for the farm, field, and greenhouse, which reported results to a central server, typically cloud-based, for storing the information.

Agricultural cyber-physical systems (CPS) are a growing research area with improving technologies. An et al. analyzed several future applications of CPS to increase the production capabilities of fields[5]. These systems gathered critical timely information about the climate, soil, and other factors affecting a crop with high granularity. The typical architecture analyzed by the authors consisted of sensor and sink nodes, a network, control center, and farming facilities. Sensor nodes gathered data and sent readings to sink nodes, which relayed them to the human-operated control center. Decisions were made at the control center and sent commands to the farming facilities to control the environment. Huang et. al. proposed a robot-based management system for managing the opening growth environments of crops[6] and incorporated a binarized neural network (BNN) for real-time analytics. BNN systems use static weights with less precision and are use bitwise operations for faster execution times and lower memory footprints[7]. A cryptographic module was included in the systems design to ensure data security in the automated environment. The robot was capable of recording crop imaging data, weather and soil information, and using these inputs to control sprinklers and agents like pesticides. Decisions were made after using a deep neural

network to identify the target crop and analyze its growth versus the ideal computerized model. The experiment consisted of a single robot to process all data, classify based on crop type, decide upon actuator states, and encrypt the information.

The proposed architecture follows similar thought processes as some prior works[3], but improves methodologies and technologies and incorporates each of the individualized subsystems into an overarching architecture with real-time predictions and automated analysis to act on environmental stimuli.

Chapter 3

System Architecture

Fig. 3.1 depicts our proposed system architecture for automated field-based tent systems for CEA. The proposed architecture comprises of the following subsystems or components: (i) Data Node, (ii) Sensor Controllers, (iii) Wireless Controller(s), (iv) Vehicle Controller, (v) Database Node, (vi) Analytics Node, (vii) Decision Node, and (viii) Actuators. The architecture is designed in a distributed control network style (Figure 1). Each location within the farming area becomes a subsystem of the overall architecture. Each subsystem is compartmentalized based upon its function. The system has the capabilities to automate irrigation, control temperature, monitor sunlight conditions, and other important factors related to crop yield via actuators and relays. The entire control system is part of a Local Area Network (LAN). Different sensor modalities communicate over the internal network by sending information to their respective controllers based on an agreed-upon protocol; I^2C devices send their readings to the I^2C controller, and UART sensors give the UART controller their data. The controllers send their readings to the Data Node to coalesce the readings and standardize them for transfer into a local database and for processing in the Analytics Node. Data stored in a local database is periodically uploaded to an off-site, backup system (i.e., cloud database) via Wireless networking. The Analytics Node takes these inputs from the Data Node and uses this input data to analyze the effect of

actuation decisions on controlled environmental conditions, and their effect on crop health. The Analytics Node provides its output to the Decision Node, which makes decisions about adjusting the controlled environment, such as modifying a heating element, maneuvering a robotic aperture device, and adjusting sensor configuration.

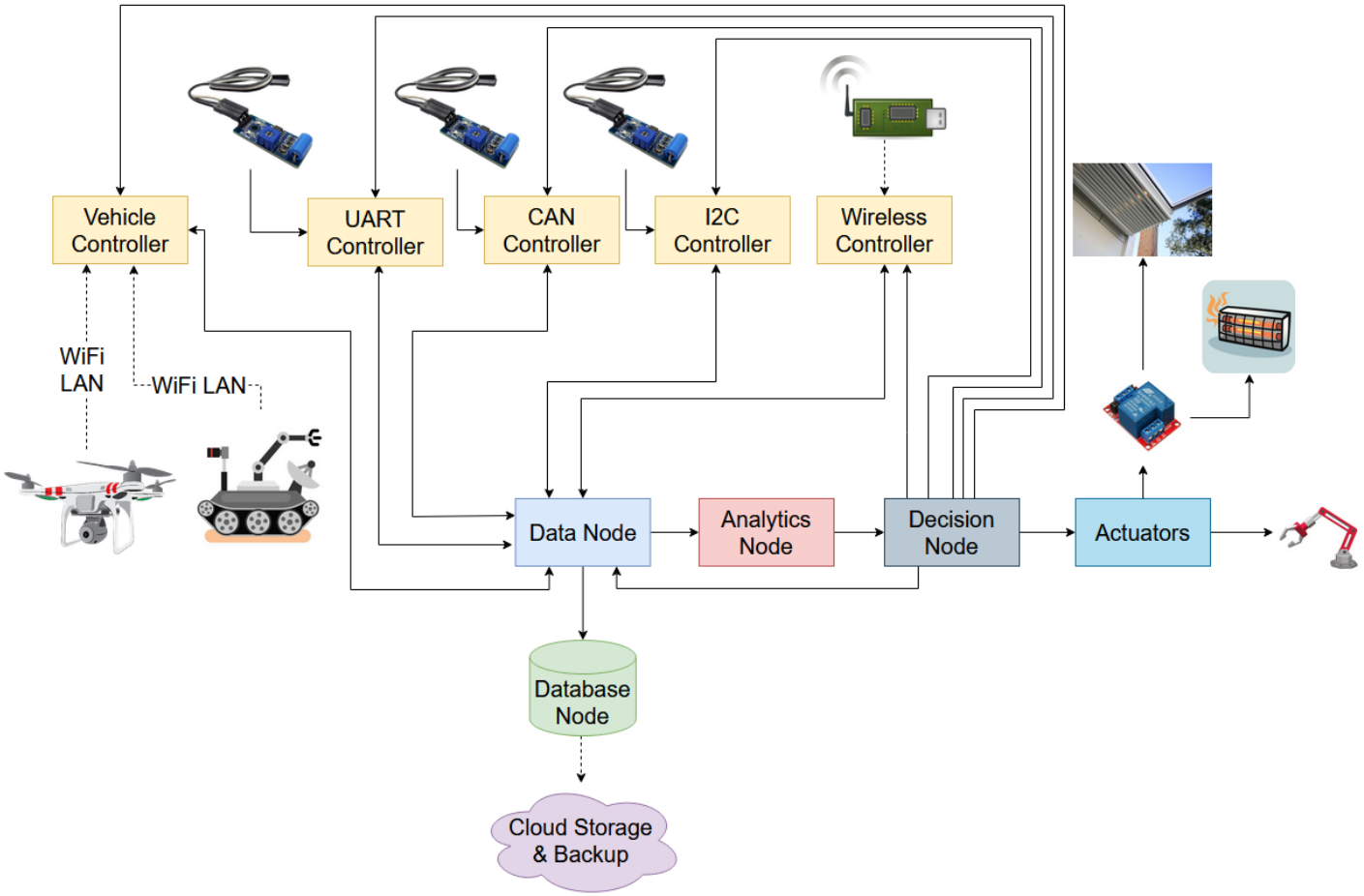

Figure 3.1: *Proposed architecture for controlled-environment agriculture.*

3.0.1 Data Node

In the Data Node, sensor readings sent from their respective controllers are aggregated and cleaned up: nil or null values are excluded, and erroneous readings are removed from the set. The node takes, as input, K sets of $(N \times 1)$ vectors of output readings from the sensor controllers in the system, with K being the number of different sensor controllers. Data is packaged into a nested JSON format, with readings indexed by the type of sensor (i.e., CAN, I^2C, UART). Readings are aggregated by either including each sensor value or by averaging values of the same type (e.g., temperature, humidity) before validation. After multiple readings from all controllers are validated and combined into a single JSON document, then they are sent over the LAN to the Analytics and Database nodes in transmission control protocol (TCP) packets. The packets consist of a vector of size $(K \times N) \times 1$ containing the coalesced readings. The Analytics Node then acknowledges receipt of the data packet before the Data Node can aggregate more readings. Any subsequent readings received by the Data Node are kept in a buffer pending processing. This ensures that processing is available for the recent readings. We have used a buffer of size 4096 bytes, but a larger buffer size can be used if imaging sensors (or other complex data sensors) are incorporated into the controlled-environment system for crop monitoring. Once the buffer has reached maximum capacity, the sensor controllers are signaled to halt sending their packets until a certain amount of time has passed. The backoff time increases with subsequent expirations of time counters following a Binary Exponential Backoff model from network engineering. In the BEB model, once a failure (in this case, a halt signal) is detected, the amount of waiting time for the next attempt is doubled. The proposed model translates the Data Node's halt signal to the controllers as the failure[8]. Thus, consecutive halt signals induce twice the delay of their predecessors to relieve backlog in the node. The controllers stop reporting readings until the delay time is completed, after which they resume sending until they receive another halt signal.

Algorithm 1 Data Node Operations

Input: S, b, n

Output: S' - S coalesced into nested JSON

b ← binary exponential backoff time, in seconds ;

if $n \geq 256$ **then**

$r \leftarrow HALT$;

$t \leftarrow b$;

$b \leftarrow b * 2$;

send r, t to all Sensor Controllers ;

else

$Q \leftarrow S$;

end if

for $Q_i \leftarrow Q_1$ to Q_n **do**

for $S_i \leftarrow Qi_j$ to Qi_n **do**

if Si is not NULL **then**

$P[i] \leftarrow S_i$;

end if

end for

end for

P' ← remove erroneous readings from P ;

S' ← package P' into JSON format ;

return S'

3.0.2 Sensor Controllers

One sensor controller exists for each different protocol that the sensors use. Each sensing instrument is directly managed by its respective Sensor Controller, which makes configuration

changes and polls sensors for their data readings. New sensors are registered with the system before being able to communicate. Once new sensors join the network, they must send a registration request to the controller, who can accept or decline communications based on physical system requirements, like power consumption, or from software requirements, such as schedule feasibility. Sensors are polled, according to their available sampling rates and the experimental requirements, for their raw readings by the controller, which then decodes and reduces them to send to the Data Node. Power consumption is minimized by using the appropriate sleep policy in each sensor controller. Up to N different sensor readings are taken as input and reduced to a $(N \times 1)$ vector output. Reduction is typically an average computation but is specified by the experimental requirements and can include functions such as `average`, `max`, `min`, or `sum`. Multiple controllers can send their results to the Data Node in a normal operation scenario for processing; however, if network link bandwidth is limited, then controllers are ranked by priority first according to the number of active sensors that they manage, and second by their update frequency (i.e., 1Hz is less prioritized compared to a sensor operating at 10Hz). As an example, given a network consisting of five CAN sensors operating at 10Hz intervals, three I^2C sensors operating at 400 kHz, and two UART sensors operating at 1Hz, the CAN sensors are top priority, I^2C second, and UART last. Each Sensor Controller parses configuration instructions from the Decision Node and ensures they are valid commands before transmitting them to the specified sensors. Any configuration changes requested by the Decision Node are carried out by the individual controllers directly without requiring the other controllers to receive any extraneous messaging. Upon receiving a set of M configuration changes, the controller outputs those M modifications to the individual sensors attached; thus, M is less than the total number of sensors in the entire system.

Algorithm 2 Sensor Controller Operations

Input: C

Output: S' - aggregation of sensor readings

R $\leftarrow$ register sensors that are connected on the bus ;

for $R_i \leftarrow R_1$ to R_n **do**

  $P[i] \leftarrow Ri$;

end for

r, $t \leftarrow$ send P to Data Node ;

if r is HALT **then**

  sleep sensors for t ;

  sleep controller for t ;

else

  $S' \leftarrow$ aggregate P by user-defined method ;

end if

if $\#C \geq 1$ **then**

  for $C_i \leftarrow C_1$ to C_n **do**

    $v \leftarrow$ check if C_i is a valid command ;

    if v is true **then**

      apply configuration to sensors ;

    end if

  end for

end if

return S'

3.0.3 Wireless Controller

The Wireless Controller is responsible for configuring and communicating with connected wireless sensors. Each sensor acts as a transmitter, where it reads data and transmits it back

to the controller for reading. The controller itself is a transceiver that reads the data as well as sends configuration changes requested by the Analytics Node to the affected sensor units. The wireless controller is also responsible for sending the data to the Data Node for coalescing and further transmission. Communication is done on the standard 2.4 GHz wireless frequency band; nevertheless, sensors could be programmed to operate at a lower frequency for power savings. Sensors follow a previously proposed adaptive sleep methodology based on network conditions to reduce as much power as they can[9].

3.0.4 Vehicle Controller

The Vehicle Controller communicates with and manages the autonomous vehicles in the system. Detected vehicles within the range of WiFi or Bluetooth signals are managed by creating a subnet-like architecture. Robots connected to the infrastructure can automate crucial tasks within the farm and automatically report back via the Vehicle Controller. The controller itself acts as the router for the subnet. To prevent communication issues, the controller follows the BEB algorithm as described in the Data Node section. Failures to communicate with a vehicle are recorded, and the controller attempts to periodically reconnect the vehicle.

3.0.5 Database Node

The Database Node receives the aggregated data in a nested JSON format from the Data Node. It takes the $(K \times N) \times 1$ vector of readings as input from the Data Node, and outputs a single boolean denoting success or failure from interacting with the database. Any sanitization occurs in the Database Node to deal with unusual nonprinting characters, irregularities not caught and handled by the Data Node, or problems that would inhibit insertion into the database. A NoSQL schema is used for implementation due to being plug-and-play with different numbers and types sensing components. Periodic backups would be uploaded to cloud-based database once per day at a minimum. Data is archived for

recordkeeping and for the potential use in future machine learning model training and tuning.

3.0.6 Analytics Node

The Analytics Node receives the data readings from the Data Node, in JSON format. Users instruct this node of the functions to perform while the robots and other automaton execute the tasks without requiring human interaction. Before processing can occur, the JSON data is extracted and placed into a vector format. An ANN is located at this node that takes the vector of data and performs the analysis to provide insights into the relationships between different controlled-environment and actuators' settings; however, not much work was done regarding the network since the overall system design was the primary focus. Mathematically, the output of a single neuron of an ANN with n neurons in the previous layer can be expressed by

$$y = f\{(\sum_{i=0}^{n} w_i * x_i) + b\}$$

where each input has a weight factor w_i that represents its perceived influence on the output, and a bias b is added to the outcome to shift the result towards a more accurate and flexible model. The overall function on the input, f, is the activation function that limits the output of a neuron to specific ranges; typically, the `sigmoid`, `hyperbolic tangent`, `rectified linear unit` (RELU), and `softmax` functions are used. The model is trained and tuned based upon previous readings recorded by the Data Node. Most of the training comes from archived data, while tuning is done periodically with live data for improvements. Data from previous harvest years can be used to train the neural network to both predict maximum yield and maintain the conditions to achieve the best results for that field in real-time.

Based on the input data, the Analytics Node conducts analysis of various actuator settings and their effect on controlled-environment variables (e.g., temperature, humidity). The Analytics Node also classifies the controlled-environment conditions in different levels (e.g., severity of frost, temperature stress, humidity, etc.) based on the observed sensor data and

actuator settings. The Analytics Node also provides predictions on the effects of different actuator settings, which is provided as input to the Decision Node to assist in decision-making. Results from the neural network can help identify important factors affecting crop health and quality, and help improve them through constant adjustments that are made by Decision Node based on the analysis. The system periodically performs this cycle of sense-analyze-direct at a defined time interval to let the environment properly respond to the prior changes (e.g., actuator settings) and help prevent noisy and erroneous readings from affecting the system.

3.0.7 Decision Node

The Decision Node receives the analytics from the Analytics Node, and take decisions, such as sensor and actuator configurations. The decision node produces generic commands, such as "increase temperature". These commands are analyzed to extract the meaningful configuration changes that need to be made and convert them to the appropriate syntax for each respective controller and actuator. Bounds checks and other data validity tests can be performed in this subsystem. The Decision Node sends each specific configuration change to the appropriate sensor controller as a packet which is then consumed. Actuators are changed by the Decision Node itself by altering the relays connecting electrical components or maneuvering apertures via servo or another instrument. Decisions are also sent to the Data Node for archival purposes.

Algorithm 3 Decision Node Operations

Input: D

Output: C - changes to actuator and sensor state

for $D_i \leftarrow D_1$ to D_n **do**

d, $e \leftarrow$ convert D_i from decision to direction and environmental variable

send d, e to the appropriate governing sensor controller

end for

3.0.8 Actuators

Actuators can be any device ranging from a robotic aperture used to adjust shutters for sunlight level modulation to a propane heater for controlling temperature. Actuators receive the commands from the Decision Node and alter the relays connecting electrical components or adjusts the apertures (controls) within the system to govern the environment. For example, propane heaters, motorized blinds, and LED lighting systems can be controlled to affect the environment in different ways. Additionally, temperature systems, sunlight controls, and some irrigation systems can be controlled to further automate and regulate the system.

Chapter 4

Control and Actuation

Control is modularized and localized into specific sections of the architecture. Each sensing instrument is directly managed by its respective Sensor Controller, which makes configuration changes and polls sensors for their data readings. Each Sensor Controller parses configuration instructions from the Decision Node and ensures they are valid commands before transmitting them to the specified sensors. Erroneous sensors are handled by the Sensor Controller through direct contact and by reporting errors for storage in the Database Node; simple errors that require a hardware restart or software reconfiguration are directly performed by the controller with minimal reporting, while complex errors that may require human interaction are reported immediately to the Database Node. Sensor configuration instructions and actuator changes from the Decision Node are first determined at the Analytics Node by the trained neural network and then relayed to the Decision Node. The system periodically performs this cycle at a defined interval to let the environment properly respond to the prior changes and help prevent noisy and erroneous readings from affecting the system; environmental factors include the size of the area managed by the system, the desired configuration, and the rate of change that a configuration/sensor change exhibits on the environment. Information in the database is directly controlled by the Data Node: data is cleaned up, formatted, and coalesced before being transmitted to the Database Node for

insertion.

The control loop begins with sensors reading environmental data. Sensor Controllers poll their attached sensors to retrieve the readings and aggregate them; aggregation method is defined by the user and can be functions such as average, summation. The controllers send the aggregated readings to the Data Node. The Data Node ensures that processing is available for the recent readings; if the node is working on a full buffer of readings, then all Sensor Controllers are notified to stop sending messages for a certain amount of time governed by BEB. At the Data Node, the multiple readings from all controllers are validated and coalesced into a single JSON document before being sent to the Database and Analytics Nodes. Within the Database Node, the database system is managed and periodically backed up to a cloud service. The Analytics Node makes a decision based on the readings and informs the Decision Node of changes that the system requires to maintain the environment. Finally, the Decision Node directs the actuators and Sensor Controllers directly with commands for changing their states before the loop begins anew.

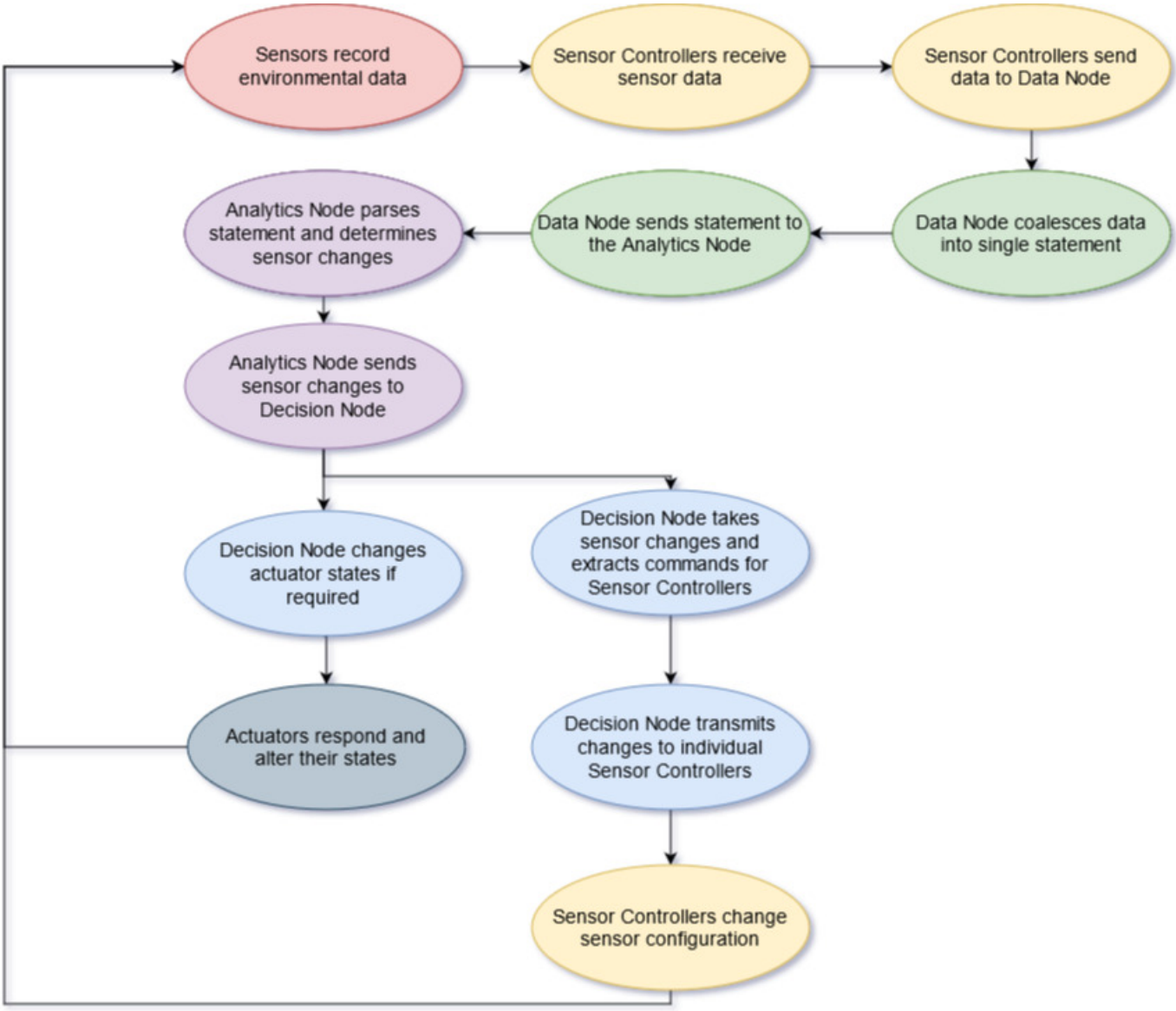

Figure 4.1: *System flow diagram.*

Chapter 5

Commercial Options

Options for sensor controllers require interfaces to their respective sensors. A cost-effective, efficient, versatile hardware solution for the sensor controllers is a Raspberry Pi. These devices have hardware interfaces for I2C, UART, and SPI, with attachments available for interfacing with non-traditional microcontroller connections like CAN bus. Table 5.1 lists several varieties of Raspberry Pi that are available, depending on the requirements of the system. Each model has improved memory and processing capabilities from the previous generation. All models can communicate over I2C, SPI, and UART, but are limited: the Pico can only communicate with 2 I2C devices while all others can handle 127 devices, and each model can only communicate with a single UART module directly connected.

For the standard sensor controllers, the Raspberry Pi 4 Model B with 2GB is recommended. The processing capabilities are plenty for general work, the power consumption is average (2.5A maximum), and the memory is more than sufficient. However, the vehicle

Table 5.1: *Raspberry Pi models available for purchase (from raspberrypi.org)*

Component	Cost	Wireless	Memory	Processor
Pi Pico	$5	No	264kB SRAM, 2MB Flash	Dual-core ARM Cortex M0+
Pi Zero W	$13	Yes	512 MB	BCM 2835
Pi 3 Model B+	$47	Yes	1GB	Quad-core ARM Cortex-A53
Pi 4 Model B	$47	Yes	2-8 GB	Quad-Core ARM Cortex-A72

controller would require additional processing power if imaging data were being processed; in this case, the Pi 4 Model B would be a starting point before extending into NVIDIA Jetson modules or equivalently powerful embedded devices: memory and processing capabilities become paramount.

Many common environmental sensors communicate over the I2C protocol, with some specialized sensors for air quality analysis using the UART protocol. Devices on the I2C bus take advantage of the high-speed mode, which can operate between 100kHz and 400 kHz. Some sensing units, like TE Connectivity's Ambimate MS4 Multi-Sensor Module, combine multiple different sensors onto a single board[10]. I2C devices, while operating at high frequencies, are limited in wire lengths from sensor to controller: the longer the signal must travel, the slower the bus must operate. UART devices CAN devices require additional hardware for use in a Raspberry Pi system and are limited to two channels[11]. Table 5.2 lists some possible options for sensors that record air temperature and humidity. For the Honeywell HIH series, as the humidity accuracy threshold decreases, the cost increases; otherwise, the base model is sufficient for most applications; however, the chip itself is not plug-and-play into a Raspberry Pi, as it requires custom leads or harnessing to properly secure. The SeeedStudio Grove AHT20 sensor can connect its data lines to the Grove Hat (also available) for easier integration. The power consumption is lower than the Honeywell equivalent, but sacrifices temperature resolution, operating temperature, and an address on the I2C bus by factory settings. TE Connectivity's HTU31D falls between the HumidIcon and AHT20, with a wider range of operating temperature, improved temperature resolution and accuracy. The module itself requires nontrivial wiring or pin arrangements as it ships from the factory as a chip without breadboard. Airmar produces a WeatherStation that runs on the CAN bus and outputs multiple data elements, such as GPS, wind speed and direction, barometric pressure, air temperature and relative humidity. The Airmar WX150 WeatherStation was included for protocol comparison. Most options are limited to I2C and SPI, with more expensive (and feature-rich) devices using CAN or Serial RS-232. The I2C devices operate at higher

sampling rates and consume less power than the CAN bus counterpart. In this comparison, the WX150 WeatherStation is worse in all aspects, with higher power consumption, stricter operating environment temperature ranges, less humidity accuracy, lower temperature resolution, lower temperature accuracy, and a much lower sampling rate. The device uses either Serial RS-232 or the CAN bus for communications.

Table 5.2: *Sensor options for recording air temperature and relative humidity*

Model	Operating Temperature (C)	Power Consumption	Humidity Accuracy (+/-)	Temperature Resolution (C)	Temperature Accuracy (+/-)	Sampling Rate
HumidIcon HIH Series	-40 to +100	1uA sleep, 650uA active	2% to 4.5%	0.025	0.5	100kHz to 400kHz I2C, 50 to 800kHz SPI
Grove AHT20	-40 to +85	0.25uA sleep, 23uA active	2%	0.01	0.3	1 to 400kHz I2C
HTU31D	-40 to +125	0.40uA sleep, 1.04uA average	2%	0.016	0.2	100kHz to 400kHz I2C
WX150 WeatherStation	-25 to +55	75mA	5%	0.1	1.1	1Hz to 10Hz CAN, RS-232

Chapter 6

System Applications

The proposed architecture for automated field-based tents for CEA allows for a variety of experimental uses. A few of the applications of the proposed architecture are discussed below.

Crop Phenotyping: Crop phenotyping applications could benefit from the proposed system. Different sensing components and subsystems can be integrated easily into the architecture by the design of the sensor controllers which would allow multiple phenomenon to be sensed. The architecture provides custom data logging capabilities tailored to the designer's or user's needs, and allows interchangeability of components to best suit the experiment. Paired with the Database node, collecting and analyzing physiological phenomenon are streamlined, autonomous, and are non-invasive to the crop itself if appropriate sensors and actuators are chosen and installed within the operating area. Image acquisition can be done by a UAV and processed by the Vehicle Controller to capture the interaction between crop and environment[12]. Crop growth stages can be identified via image and paired with the environmental readings for further insights into physiological effects.

Controlled-Environment Agriculture: Current academic research focuses upon individualized, complex systems for monitoring singular variables or interactions between the crop and environment, as well as larger-scale, IoT-based smart farming systems with multiple individual embedded devices scattered throughout the area. Castañeda-Miranda et. al. designed an IoT system to accurately predict frost levels in a greenhouse and engage an anti-frost irrigation system to prevent damage and crop disaster[3]. The authors utilize an ANN to predict the greenhouse temperature, which is then input into a fuzzy logic system to determine the appropriate amount of anti-frost to apply to maintain the thawed environment. The fuzzy system incorporates the frost weather index and places the ANN output into one of several bins based on the potential predicted hazard from the frost. By design, the system is able to power itself by using solar pannels to charge the batteries connected to the embedded electronics. The proposed architecture can incorporate a system such as this by integrating the appropriate controller(s) and communication protocols to create a larger, overarching smart farm system. Farooq et. al. surveyed the role of IoT in agriculture for smart farming incorporation[4]. They noted that agricultural trends were individualized systems for the farm, field, and greenhouse; these reported results to a central server, typically cloud-based, for storing the information. However, the trends do not incorporate more complex computing components, like neural networks and machine learning, and remain cemented in separating data collection, analysis, and real-time monitoring. The proposed architecture follows similar thought processes but incorporates each of these individualized subsystems into an overarching architecture with real-time predictions and analysis to act on environmental stimuli. Each location within the farming area becomes a subsystem of the overall architecture. The system has the capabilities to automate irrigation, control temperature, monitor sunlight conditions, and other important factors related to crop yield via actuators and relays. Power consumption is minimized by using the appropriate sleep policy in each sensor controller. Data from previous harvest years can be used to train the neural network in the Analytics Node to both predict maximum yield and maintain the conditions

to achieve the best results for that field in real-time. Results from the neural network can help identify important factors affecting crop health and quality and improve them through constant adjustments from analyses performed. Robots connected to the infrastructure can automate crucial tasks within the farm and automatically report back via the Vehicle Controller. Users instruct the system via the Analytics Node of the functions to perform while the robots and other automaton execute the tasks without requiring human interaction.

Precision Agriculture: Precision agriculture can benefit from an automated, self-learning system in tasks ranging from variable-rate Nitrogen/nutrient application (VRNA) to yield prediction improvement. Asebedo and Mengel confirmed in their studies on algorithms for optimal Nitrogen application amounts that more data was required for sufficient model accuracy[13]. Nitrogen application is a crucial step in the farming process for optimal crop growth, yield, and quality. Previous agronomic research has overwhelmingly shown that VRNA is the optimal method for crop fertilization as compared to blanket Nitrogen application within a plot. Fields are thus sub-divided, or real-time algorithms are developed, to apply differing levels of Nitrogen in different sub-fields. The primary statistic used to determine the optimal Nitrogen amount to apply to each section, Red normalized difference vegetation index (NDVI), is a singular data point. While being one-dimensional, tremendous success has resulted from analyzing and categorizing plots based on Red NDVI thresholds. However, other data values, such as weather conditions, soil information, infrared and thermal imaging may not be considered in many real-time calculations and models, which the architecture provides integration possibilities for. For example, in Winter Wheat, the tillers, head size, and grain filling are used to determine the yield and occur at different points in the Feekes scale[14]. The Feekes stages 1 to 5 occur when the plant is below the soil level, which mitigates pest and other above-ground environmental issues. At Feekes 7, Nitrogen mineralization is easier to assess and evaluate, but has the potential for tiller abortion due to Nitrogen stress[13]. Each stage can be categorized and visually identified as the plant reaches maturity. An immedi-

ately straightforward solution towards reducing Nitrogen stress is to examine and identify the plant growth stages using the imagery available and incorporate these into the real-time model to improve Nitrogen decisions. Temperature and day length are determining factors for how quickly the crop moves through the stages. Sensing instruments for these values can be easily integrated into the system, and relationships between them can be determined by the neural network within the Analysis Node.

Table 6.1: *Feekes Growth Stages in Wheat.*

Feekes Stage	Visual Description
1	Germination period to the first emerged leaf
2	Tillers become visible on the stem (new shoot that originates underground off main stem)
3,4	Tillers are formed and generated
5	Strongly erect leaf sheath, plant has upright appearance
6	First node is visible (bump on the tillers)
7	Second node is visible
8	Flag leaf is visible (last leaf begins emerging)
9	Flag leaf is completely emerged and visible
10	Head/spike is fully developed and visible
11	Ripening

Chapter 7

Case Study in HNT Experiment

We have developed and implemented a smaller version of the proposed architecture to study of the effects of high nighttime temperatures (HNT) on winter wheat in Kansas[15]. The experimental thermostat controller system is installed in custom designed tent structures that are 9.1 meters wide, 14.6 meters long, and 4.4 meters tall. Each structure is placed over eight blocks of 40 rows of different winter wheat varieties. The architecture is simplified and split into three pairs of tents, with one control and one heat tent per pairing as depicted in Fig. 7.2. Heat tents consist of a propane heater connected to the controller, a Raspberry Pi model 3B, via relay, which converts the 5V input signal from the controller to the 250V AC required to engage the heater, along with six MCP9808 temperature sensors[16], one DS3231 RTC module[17] and one MHZ-19 carbon dioxide sensor[18]. Control tents contain a Raspberry Pi model 3B and the same sensors. Their purpose is to monitor and record the ambient temperature for later use in the heat tent decision-making.

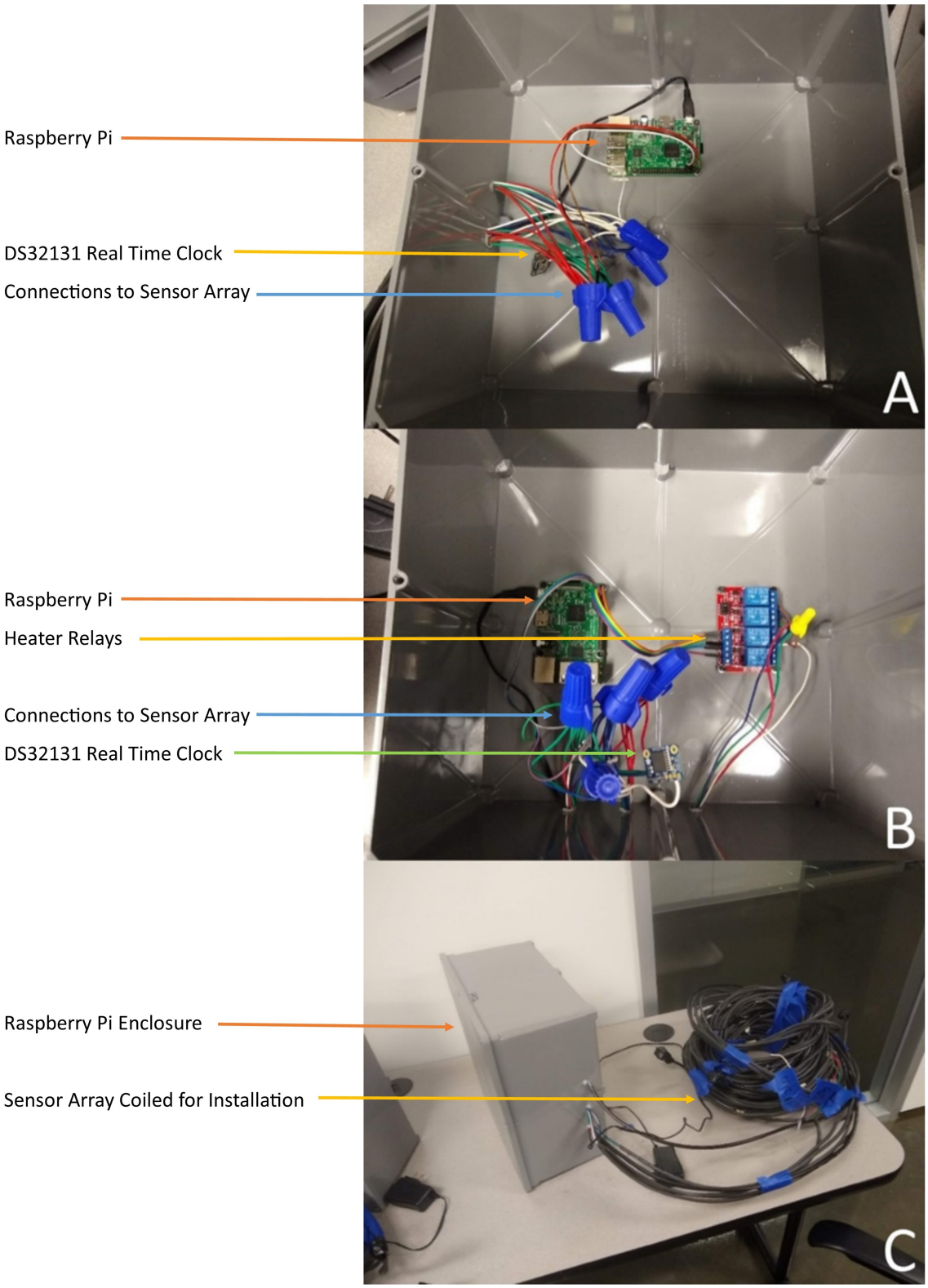

Figure 7.1: *Hardware configuration for both Heat (A) and Control (B) Tents. The sensor array (C) is also shown.*

Each pair is connected over a wireless LAN and managed by the control tent, which acts as the router. Each tent records the averaged indoor temperature and the carbon dioxide reading. The control tents transmit their readings to the heater tent for analysis, which are treated as the outdoor/ambient temperature. The heat tent, upon receipt of the readings, averages its recorded temperature and compares it to the received readings: above a 4°C threshold causes the controller to disengage the relay, and below 4°C engages the relay. The goal is to maintain a 4°C warmer temperature in each heat tent than the control tent temperature for the duration of the experiment. The system is programmed to read temperature data, decide on the relay state, log sensor and actuator data to file, and sleep. This cycle repeats once per minute to allow the change in relay state to propagate through the environment.

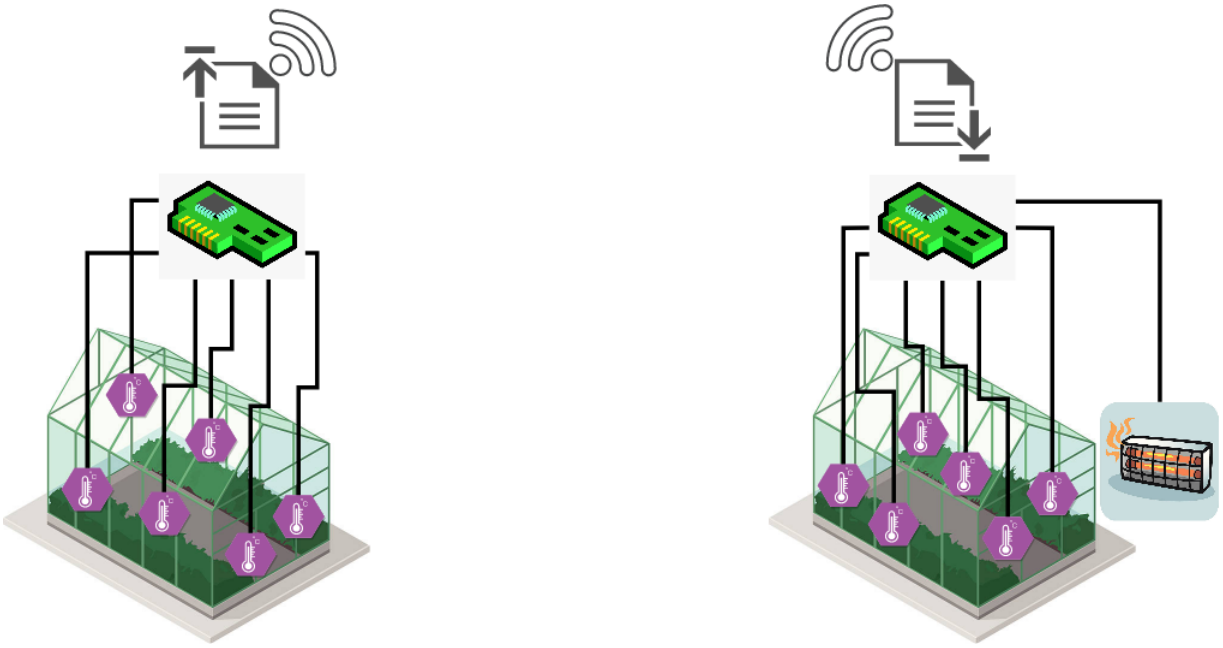

Figure 7.2: *Pairwise tent system. The tent without a heater transmits its average indoor temperature reading to the heated tent for decision-making.*

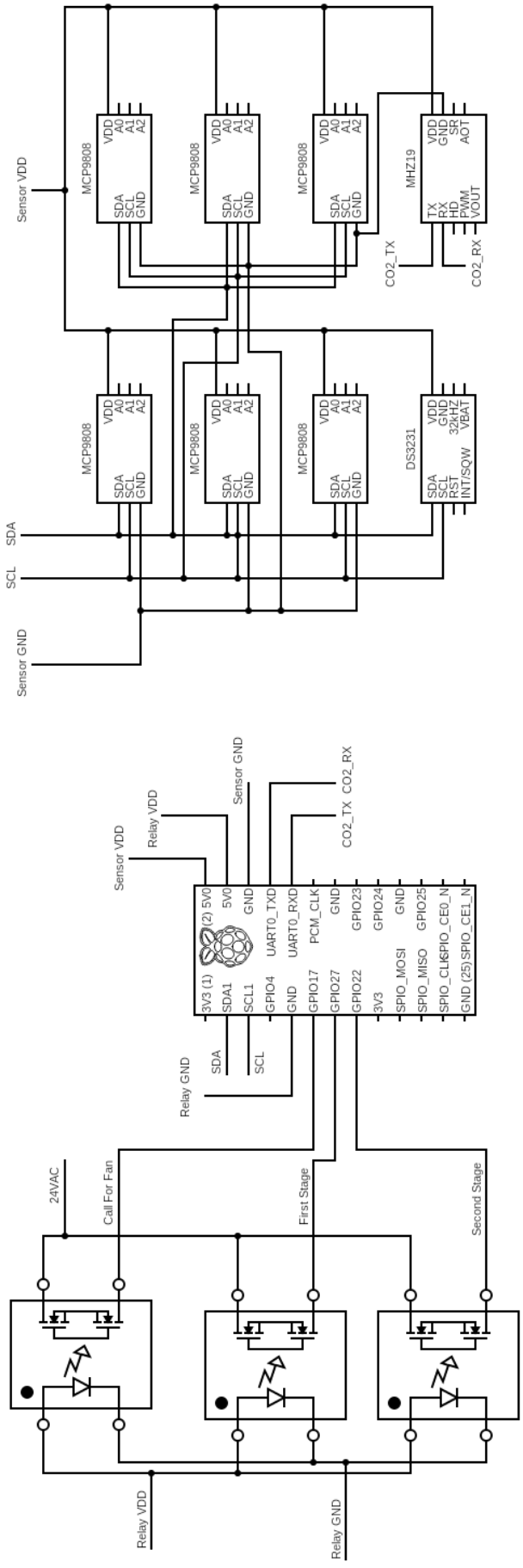

Figure 7.3: *Circuit diagram for Raspberry Pi controllers. Relays were only present in Heat Tents.*

The circuitry is connected as shown in Fig. 7.3 and is straightforward. Each MCP9808 temperature sensor, along with the DS3231 RTC module, is connected to four pins on the Raspberry Pi: SDA, SDL, VDD, and GND. These correlate to pins 3, 5, 2, and 6 on the Pi using the BCM labeling scheme. The carbon dioxide sensor shared the VDD and GND lines but use the UART protocol and require the TX and RX pins of the Pi instead of the SDA and SCL connections. The TX and RX pins correlate to pins 10 and 6, respectively, on the Pi itself. Relays for the Heat Tents shared VDD and GND with each other, but were isolated from the sensor lines by connecting them to the 4 and 9 pins. Each relay required a signal connection to change its state, and those were connected to the Pi's general purpose input/output (GPIO) pins. The first relay, which called for the heater's fan to engage, had a signal connection to pin 11. The second relay, which called for the first heat stage of the heater, was connected to pin 13. Finally, the third relay, which called for the second heat stage, was connected to pin 15. The heater's operation relied on all three signal pins simultaneously being pulled to VDD due to manufacturer safety features and electronic requirements.

Upon startup, the system detects the type of sensors connected and initializes the main controller for interfacing with the thermostat system. Two options for the main controller exist for the system: one for the heat tents, and one for the control tents; the system, at boot, selects the correct one. Each controller initializes the sensors, reboot error counter, and initiates system logging for diagnostic purposes. Once initialization is completed, the main logic loop begins with calibrating the CO_2 sensor to the current environment. Next, the MCP9808 sensors that were detected previously are polled on the I^2C bus by address with the exclusion of user-defined reserved addresses for the clock module and erroneous addresses. Any errors that occurred are logged to file and the error counter is incremented. Once a programmed level is reached with the error counter, the system automatically restarts itself to resolve the hardware issue. A programmed number of maximum restarts prevents the system from continuously restarting in order to keep the controllers online for sending

data and to maintain the HNT environment on the crops even if the temperature threshold is exceeded. The temperature of the heat tents is compared to the control tents and used to determine the relay status, which is engaged by the Raspberry Pi's GPIO pins. Each step in the process is logged for troubleshooting and archival purposes.

Both the heat and control tents act as a fusion of several of the components within the proposed architecture. They contain the following components, which are present indirectly in software as code chunks or Python classes: Data Node, Analytics Node, Decision Node, and Sensor Controller. Each tent reads the data from their sensors. In our experiments, the Sensor Controllers do not send configuration commands to the sensors, instead the default configuration and error resolution mechanisms are employed to a satisfactory level. Control tents send their data to their heat tent partners for analysis, similarly to the Analytics Node. There is no neural network present, and the decision was a choice based on a linear model, carried out by the controller itself. Data is logged to an Excel spreadsheet file and not imported into a database. The copies of the data are stored locally via replication on several different disk drives. Instead of a dynamic sleep schedule, a static minute of sleep is used in our experimental setup. The selected sleep interval is sufficient to allow the heat change to propagate through the system. The experimental implementation does not have the capabilities for image and video processing. Additional computing hardware is required for these tasks, and a redesign of the pairwise architecture is recommended.

In our experiments, the system was able to achieve a controlled environment of, on average, 3.773 °C warmer than the control areas for the duration of the heat stress as depicted in Fig. 7.4. The three respective heat tents (H1, H2, H3) were analyzed independently. Heat loss was primarily due to gaps in the plastic walls used to seal the tents. The inaccuracies in sensing implements also occurred but were within the manufactured tolerance of 0.25 °C. Heat tents 1 and 3 achieved greater than +3.8 °C temperature while heat tent 2 exhibited +3.6273 °C temperature primarily due to early relay activation issues that were resolved with replacement hardware. Heat tent 2 would engage the relay, which in turn was supposed

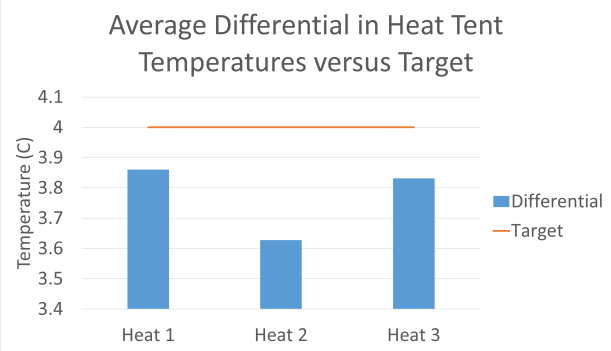

Figure 7.4: *Average recorded temperature differential in heat tent combinations.*

to turn the heater on. However, the relay was faulty and would engage only a fraction of the time. This caused the heater to remain offline for extended periods and not reach the 4 °C threshold required to maintain the heat stress. After identifying and diagnosing the fault to the relay, replacement relay sets were installed, and the issue was resolved.

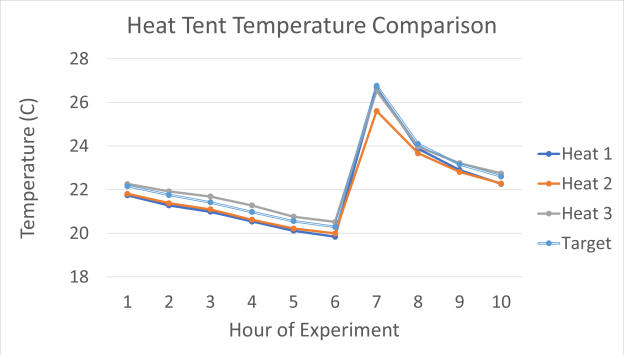

Figure 7.5: *Actual recorded temperatures in each heat tent compared to the target desired temperature.*

Chapter 8

Conclusions

While sensor systems design in agriculture are sufficient for individual, specialized experiments, there is a growing demand for an overarching architecture that is versatile, and can observe, analyze, and control multiple environmental conditions. Agronomic experiments require, and rely on, a high quantity of data that current experimental architectures are not equipped to collect, manage, and analyze in real-time. Agricultural models can be improved with additional data and sensing implements and require a solidified architecture and framework to succeed. The proposed design packages components based upon role and can accommodate a variety of sensing instruments, actuators, and autonomous robots with little effort. The proposed architecture can be utilized for a variety of applications, such as controlled-environment agriculture in field-like settings, crop phenotyping, precision agriculture, and experimentation for agricultural and agronomical research. We have developed and deployed a prototype of the proposed architecture, which demonstrates that the proposed architecture can monitor and control the nighttime temperature stress as well as other environmental conditions for winter wheat.

Bibliography

[1] About CEA. URL https://cea.cals.cornell.edu/about-cea/.

[2] Lighting, Sensing and Controls. URL https://glase.org/benefits/lighting-sensing-and-controls/.

[3] Alejandro Castañeda-Miranda and Victor M. Castaño-Meneses. Internet of things for smart farming and frost intelligent control in greenhouses. *Computers and Electronics in Agriculture*, 176:105614, 2020. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2020.105614. URL https://www.sciencedirect.com/science/article/pii/S0168169919307148.

[4] Muhammad Shoaib Farooq, Shamyla Riaz, Adnan Abid, Kamran Abid, and Muhammad Azhar Naeem. A survey on the role of iot in agriculture for the implementation of smart farming. *IEEE Access*, 7:156237–156271, Oct 2019. doi: 10.1109/access.2019.2949703.

[5] W. An, D. Wu, S. Ci, H. Luo, V. Adamchuk, and Z. Xu. Chapter 25 - agriculture cyber-physical systems. In Houbing Song, Danda B. Rawat, Sabina Jeschke, and Christian Brecher, editors, *Cyber-Physical Systems*, Intelligent Data-Centric Systems, pages 399–417. Academic Press, Boston, 2017. ISBN 978-0-12-803801-7. doi: https://doi.org/10.1016/B978-0-12-803801-7.00025-0. URL https://www.sciencedirect.com/science/article/pii/B9780128038017000250.

[6] Chun-Hsian Huang, Po-Jung Chen, Yi-Jie Lin, Bo-Wei Chen, and Jia-Xuan Zheng. A robot-based intelligent management design for agricultural cyber-physical systems. *Computers and Electronics in Agriculture*, 181:105967, 2021. ISSN 0168-1699. doi:

https://doi.org/10.1016/j.compag.2020.105967. URL https://www.sciencedirect.com/science/article/pii/S0168169920331720.

[7] Taylor Simons and Dah-Jye Lee. A review of binarized neural networks. *Electronics*, 8 (6):661, 2019. doi: 10.3390/electronics8060661.

[8] Arvindpdmn Sangeetha-Prabhu. Binary Exponential Backoff, Jul 2020. URL https://devopedia.org/binary-exponential-backoff.

[9] Giuseppe Anastasi, Marco Conti, and Mario Di Francesco. Extending the Lifetime of Wireless Sensor Networks Through Adaptive Sleep. *IEEE Transactions on Industrial Informatics*, 5(3):351–365, Aug 2009. doi: 10.1109/tii.2009.2025863.

[10] Ambimate MS4 Multi-Sensor Module. URL https://www.te.com/usa-en/product-CAT-AM16-M8797.html.

[11] Two-Channel Isolated CAN Expansion HAT for Raspberry Pi, Dual Chips Solution. URL http://www.waveshare.com/2-ch-can-hat.htm.

[12] Muhammad Tariq, Mukhtar Ahmed, Pakeeza Iqbal, Zartash Fatima, and Shakeel Ahmad. Crop Phenotyping. *Systems Modeling*, page 45–60, 2020. doi: 10.1007/978-981-15-4728-7_2.

[13] Ray Asebedo and David Mengel. A Sensor Based Algorithm for Intensive Nitrogen Management, Sep 2013.

[14] What is the meaning of feekes growth stages in wheat?, Apr 2016. URL https://agcrops.osu.edu/newsletter/corn-newsletter/what-meaning-feekes-growth-stages-wheat.

[15] Nathan T. Hein, Raju Bheemanahalli, Dan Wagner, Amaranatha R. Vennapusa, Carlos Bustamante, Troy Ostmeyer, Meghnath Pokharel, Anuj Chiluwal, Jianming Fu,

Dhanush S. Srikanthan, and et al. Improved cyber-physical system captured post-flowering high night temperature impact on yield and quality of field grown wheat. *Scientific Reports*, 10(1), Dec 2020. doi: 10.1038/s41598-020-79179-0.

[16] Adafruit Industries. MCP9808 High Accuracy I2C Temperature Sensor Breakout Board, 2020. URL https://www.adafruit.com/product/1782.

[17] Adafruit Industries. DS3231 Precision RTC Breakout Board, 2020. URL https://www.adafruit.com/product/3013.

[18] MH-Z19 CO2 and Temperature Sensor, Feb 2019. URL https://esphome.io/components/sensor/mhz19.html.