computation

*Article*

# Development of Raspberry Pi 4 B and 3 B+ Micro-Kubernetes Cluster and IoT System for Mosquito Research Applications

**Zhihao Pan [1,*], Byul Hur [1,*], Kevin Myles [2] and Zach Adelman [2]**

1 Department of Engineering Technology and Industrial Distribution, Texas A&M University, College Station, TX 77843, USA
2 Department of Entomology, Texas A&M University, College Station, TX 77843, USA
* Correspondence: zhihaopan@tamu.edu (Z.P.); byulmail@tamu.edu (B.H.)

**Abstract:** Detecting infected female mosquitoes can be vital when they transmit harmful diseases such as dengue, malaria, and others. Infected mosquitoes can lay hundreds of eggs in breeding locations, and newborns can transmit diseases to more victims. Hence, gathering and monitoring climate data and environmental conditions for mosquito research can be valuable in preventing mosquitoes from spreading diseases. To obtain microclimate data, users such as mosquito researchers may need weather stations in various locations and an inexpensive, effective IoT system for monitoring multiple specific locations. We can achieve this in each location by sending microclimate data from wireless sensor end-node devices to a nearby middle-node aggregator. Each location's aggregator can send the data to a cluster, such as a customized Raspberry Pi-based cluster with Micro-Kubernetes as its distributed operating system. The applications, such as the database and web server, can be wrapped up by docker containers and deployed as containerized applications on the cluster. This cluster can store the data, available to be accessed via Android and web applications. The results of this work show that the measurement data of the specific locations are more accurate than those from nearby third-party weather stations. This proposed IoT cluster system in this paper can be used to provide accurate microclimate data for the selected locations.

**Keywords:** Micro-Kubernetes (MicroK8s); mosquito research; Raspberry Pi; cluster; Internet of Things (IoT)

check for updates

**Citation:** Pan, Z.; Hur, B.; Myles, K.; Adelman, Z. Development of Raspberry Pi 4 B and 3 B+ Micro-Kubernetes Cluster and IoT System for Mosquito Research Applications. *Computation* **2022**, *10*, 221. https://doi.org/10.3390/computation10120221

Academic Editor: Karlheinz Schwarz

Received: 12 November 2022
Accepted: 6 December 2022
Published: 16 December 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Mosquito research applications have been developed and used worldwide because they are approaches for understanding and analyzing mosquitoes' behaviors. Mosquitoes feed on other organisms' blood, mainly by biting their targets with their proboscis during an organism's moments of inattention. While ingesting their target's blood, mosquitoes can potentially transfer illnesses they carry into their target's body. Therefore, the mosquito is one of the world's most dangerous creatures because it can easily infect mammals, birds, and humans with harmful diseases, such as dengue, malaria, West Nile, and Zika [1–3]. Moreover, due to changes in temperature, humidity, wind, and other microclimate variables, viruses and other diseases can quickly spread further and mutate during viral transmission among different targets, including infected ones [4,5]. With recent advancements in the microcomputer cluster and Internet of Things (IoT) systems, mosquito research applications that use these systems to gather and track microclimate data in different locations have become indispensable for predicting, detecting, controlling, and preventing mosquito breeding and disease spreading [6,7].

Due to the invention of microcomputers such as Raspberry Pi (RPi), mosquito researchers can use an RPi-based distributed computing cluster system instead of a simple single computing system. The single computing system refers to a cloud, typically offered as a cloud computing service by companies such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform. The RPi-based cluster refers to multiple RPis grouped over a network that functions as a single computer. Each RPi is considered a node in the cluster, and with Micro-Kubernetes (MicroK8s), load balancing can distribute tasks, which are workloads, to each node.

There are advantages to using the RPi-based MicroK8s cluster system instead of a single computing system. The cluster can be scalable and flexible because new nodes can be connected to the cluster to solve the problem of an overloaded workload without changing or stopping any ongoing applications. When one node or application is down, the cluster will provide highly available and fault-tolerant services, and the other nodes will take on the additional workload of that failed node or application. Regarding cloud computing, the third-party cloud provider may share the tenant data with business partners, law enforcement agents, or governments, especially when a law enforcement agency subpoenas them for tenant data. When using a cloud, the long-term availability and dependability are unpredictable because the cloud provider or service may not exist tomorrow. Additionally, the cloud service architecture is less flexible and less secure in terms of performance [8].

Computer clusters can focus on docker containerized applications, such as databases and web services, and reduce the workload of storing data and hosting web services by using Kubernetes (K8s) [9]. Since 2015, K8s has been designed by Google to be an open-source platform that can automate containerized applications' deployment, management, and scaling across the nodes of the cluster [10]. The RPi-based cluster can use different container orchestration tools as the distribution system for load balancing, but MicroK8s was chosen in this study. MicroK8s is a lightweight version of Kubernetes (K8s), an open-source container orchestration tool that can manage containerized applications, workloads, and services. By using the cluster in the IoT system, IoT devices can connect with the cluster so the cluster can store and view the microclimate data from different locations by mosquito researchers.

In this paper, a low-cost customized service solution is presented, which is a Raspberry Pi(s) Micro-Kubernetes cluster consisting of one RPi 4 B and three RPi 3 B+s. It uses the method of one master node and multiple worker nodes. We only used four RPis as the adequate solution because the cluster can add more RPis later, and the supply chain and chip shortage in 2021∼2022 [11]. In our scenario, we simply deploy our cluster in an IoT system based on our customized Internet of Things (IoT) network and IoT edge devices, like an aggregator. The aggregator, such as the sensor hub and gateway, allows microclimate data from end nodes to be sent across the internet to our cluster [12,13]. This IoT network technique combines older work with our cluster to form a data acquisition system to allow any individual to study, predict, and prevent the spreading of mosquito-borne diseases. The work presented in this paper is part of the first author's master's thesis work in 2022, and as of this publication, the thesis is not publicly accessible as it is in the embargo period.

This paper is organized into three parts. First, we start with the proposed method. After that, this study's measurements are shown, and the advantages of the proposed method are described. Lastly, we close our paper with a conclusion and some perspectives for our proposed works.

## 2. Proposed Method

### *2.1. Iot System Architecture*

In this section, the proposed IoT system is described, and it is the architecture of an efficient end-to-end process for individuals to monitor the microclimate data from any wireless sensor end-node devices. Figure 1 represents an IoT system operated by a cluster and four aggregators (remote data stations) in an IoT network.

The remote data stations (on the right side of Figure 1) are the middle node (between the cluster and the end nodes) for gathering and transferring the temperature, humidity, pressure, light, and GPS (Global Positioning System) coordinates in the mosquito breeding locations, such as standing water and containers that hold water. However, in our case, the sensors are physically connected to the remote data stations, but originally they are connected to the wireless sensor end-node devices, which are not included in this study's scope.

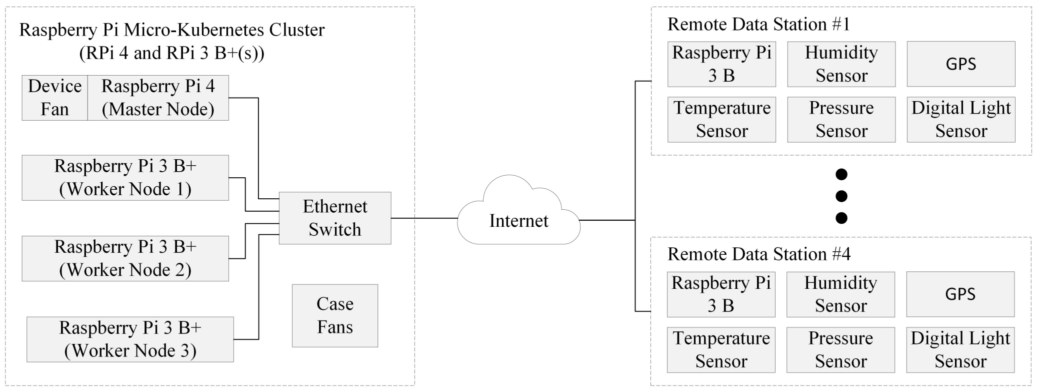


**Figure 1.** IoT system for mosquito research applications [14,15].

The task of environmental conditions and climate data collection can be challenging due to the devices' complex setup and the harsh environment. The local microclimate data can be obtained from third-party sites, but they may not necessarily accurately represent the data at a specific location [16,17]. Therefore, our remote data stations are located near mosquito breeding sites. Each station is an aggregator for periodically collecting microclimate data from various sensor nodes in the potential mosquito breeding sites. Additionally, a single RPi board is an adequate solution for each station because it can reduce the network transmission loading relief and periodically upload the microclimate data to our cluster. From others' research, individuals can also use an RPi in a fully controlled street lighting isle and human-powered vehicle [18,19].

Due to the efficiency of the IoT system, the IoT system based on cloud services has been used for structural health monitoring to increase human safety and reduce maintenance costs of the system [20]. Additionally, RPi has been used as the IoT edge device to gather climate data and upload it to the cloud services [21]. Moreover, in 2016, Keijo Heljanko designed and developed a Kubernetes cluster consisting of five RPis to act as the IoT sensor node to send pictures and temperature values to a cloud platform that used the Apache Kafka framework [22]. However, comparing our IoT system and cluster to the work done by others, our IoT system can increase the number of aggregators for monitoring microclimate variables on more mosquito breeding sites. Additionally, because of the aggregator's abilities, we can reduce latency and bandwidth costs when data are transitioning from the wireless sensor end-node devices to the cluster.

### 2.2. *MicroK8s Cluster Architecture*

In Figure 2, the architecture of the Micro-Kubernetes cluster is shown. Micro-Kubernetes (MicroK8s), a 1.23 stable version, is a lightweight and fully compliant Kubernetes (K8s) distribution system for our cluster. It can provide and simplify the usage of K8s for each RPi in our cluster. Each RPi has installed Ubuntu 18.04 LTS (Long Term Support) as the operating system because Canonical supports and features its version of MicroK8s [23]. The primary reasons for using MicroK8s are the redundancy, scalability, and reliability factors. The monotony of MicroK8s provides uninterrupted services, such as database and web service, to the users. By using MicroK8s, users can add more services by adding more RPis (worker nodes) to the cluster. With the redundancy and scalability of MicroK8s, users can depend on the cluster to collect microclimate data from each location's remote data station.

The cluster consists of one master node and three worker nodes. Each node has components that will be described. The master node (on the left) controls and manages the worker nodes (on the right) and distributes the workloads, such as hosting web services, to them. It is also responsible for orchestrating containers containing the web application and other applications on the worker nodes. The master node has a few more add-ons, including a Kube API (Application Programming Interface) server, controller, scheduler, and etcd.

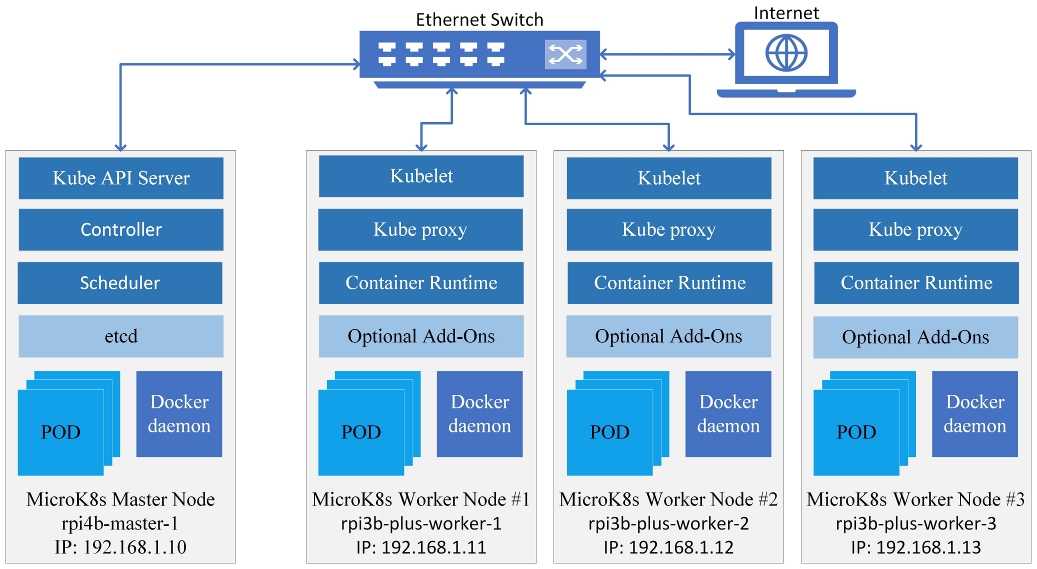


**Figure 2.** The MicroK8s architecture for the distributed system in this study [24,25].

With the use of these add-ons, users can automate the management of the cluster. The Kube API server, the front end of the cluster, allows the users, management devices, and all external communication to interact with the cluster. The controller is the orchestration's brain, which makes decisions to bring up new containers when nodes, containers, or other endpoints go down. The scheduler is responsible for distributing the workloads or containers to different nodes, including the master node. The etcd is the key value store used by MicroK8s to store all data used in the cluster in a distributed manner.

Besides the significant components from the master node, all nodes have pod(s) and a Docker daemon. A pod is a group of one or more containers, such as Docker containers, and also a single instance of an application, such as a database. Nodes can increase the number of pods across nodes to share the workloads when more users are accessing the application. The Docker daemon manages the objects, such as the Docker image, container, and network. Web and other applications can be built and run in the Docker container.

Besides the master node, there are three worker nodes. Each of them contains the Kubelet, Kube proxy, container runtime, and optional add-on(s). The Kubelet is the node agent of the worker node. It can register the worker with the Kube API server to allow itself to communicate with the master node. The Kube proxy is used as the network proxy, which maintains the network rules on the worker node. The container runtime is used to manage and support the worker node continuously in a life cycle. If the worker nodes have extra memory usage, they can add optional add-ons, such as CoreDNS, Calico, and Flannel, which extend the functionality of MicroK8s.

Figure 3 shows the front and top view of the RPi MicroK8s cluster. The enclosure is made of metal and can house four RPi boards.

Each RPi is firmly installed on the enclosure. The RPi on the left side of Figure 3 is an RPi 4 B board. The rest of them are RPi 3 B+ boards. Each RPi is connected to an Ethernet switch, which provides internet access and communication between the RPi boards. The RPi 4 B board has an external USB flash drive to store the database file. All of the database data and configuration files are stored in the USB flash drive.

The dimensions of our cluster are 20.6 × 13.6 × 16.2 cm. The back view of the cluster shows the case fans and the port that allows power cables to connect to the RPis and the Ethernet switch. Two case fans are used, and a device fan is installed on the RPi 4 B board.

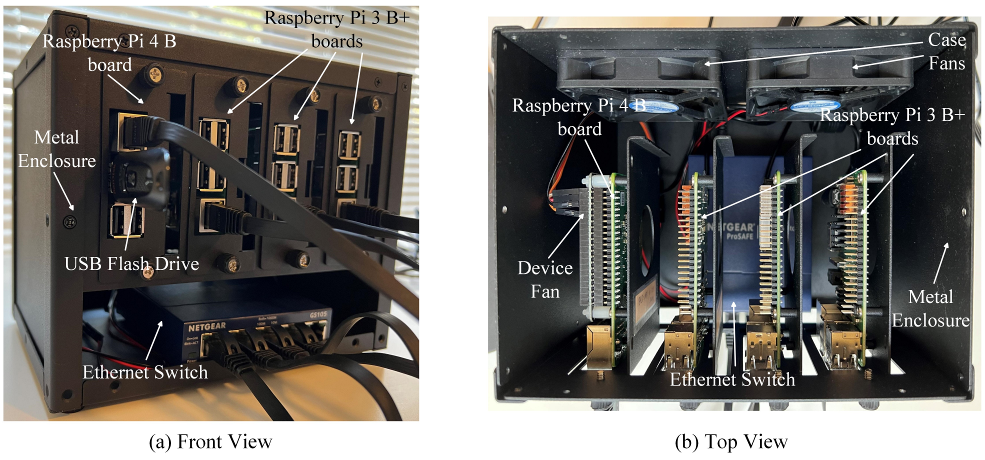


**Figure 3.** The RPi MicroK8s cluster. (**a**) Front view of the cluster. (**b**) Top view of the cluster.

## 2.3. Remote Data Station

Figure 4 shows a remote data station (aggregator). In our scenario, we used four of these and placed them in four different mosquito breeding locations. It has a 3D-printed enclosure to protect the device. Additionally, it is an embedded system with a Raspberry Pi 3 B board and sensors. The sensors are supposed to be connected to the wireless sensor end-node devices, which are not included in this study's scope. If the end-nodes are included, they can transmit microclimate data to the aggregator. In mosquito research, the microclimate data and GPS coordinates are some of the important factors for determining the method of mosquito population control. Therefore, this remote data station can currently measure temperature, humidity, pressure, and light, as well as GPS coordinates. Moreover, this system has user buttons and an RGB (Red, Green, and Blue) color indication LED (Light-Emitting Diode). Users can transfer the measured microclimate data from these remote data stations to the cluster using the MQTT (MQ Telemetry Transport) protocol.

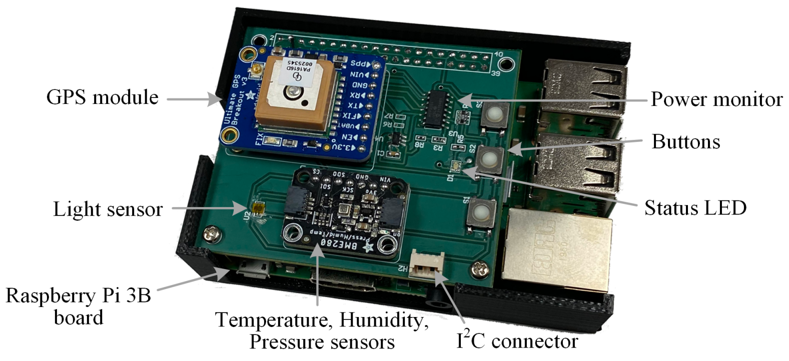


**Figure 4.** Remote data station, which is an aggregator between the cluster and the end-nodes.

## 2.4. MicroK8s Cluster and IoT System Implementation

Our cluster is formed by connecting four RPis to a network switch. Users can place it in a room with an internet connection and good airflow conditions for the operations of fans. The benefit is that users can position the cluster anywhere with those conditions. In our case, the cluster was installed and placed in College Station, TX, USA. It allows users to access the microclimate data via our customized Android and web applications. Those data are available on the database when they are sent from the remote data stations.

The flowchart in Figure 5 shows the sub-process of the measurement, which can repeatedly be running in the remote data station. Each remote data station can collect data from various sensors at about 2∼3 s. Next, it can publish the data to the MQTT (MQ Telemetry Transport) broker on the cluster. Because each station has a unique device ID (identification) number with an MQTT topic equivalent to that ID, the MQTT broker can filter each message for each connected station based on the ID. For example, if the remote data station's ID is 101, then the topic is 101. After that, the database can store the data from the MQTT broker, as shown in Figure 6, but the exact GPS locations are hidden in the figure. The MySQL database, version 8.0, is used in our scenario because the stability of the database is an essential factor in the IoT system. The database needs to store a large amount of data coming from the remote data stations, so when MySQL compares to MongoDB and other databases, MySQL is more stable for storing the data [26,27].

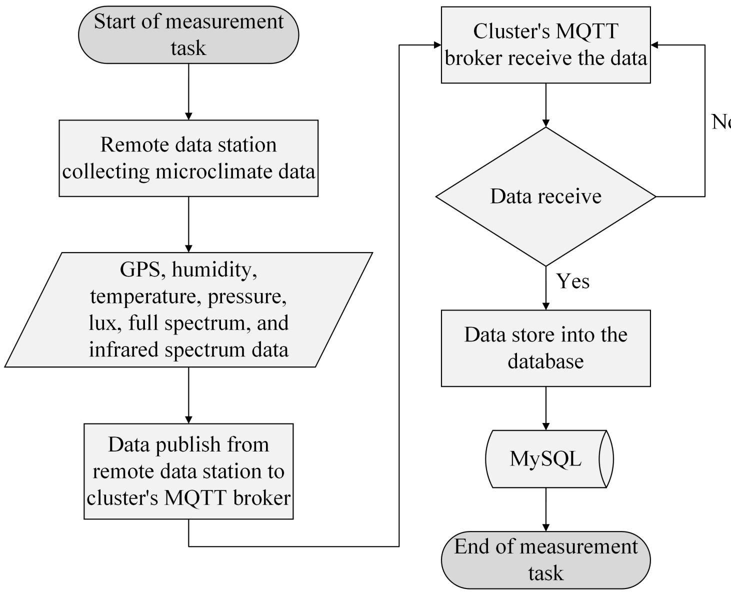
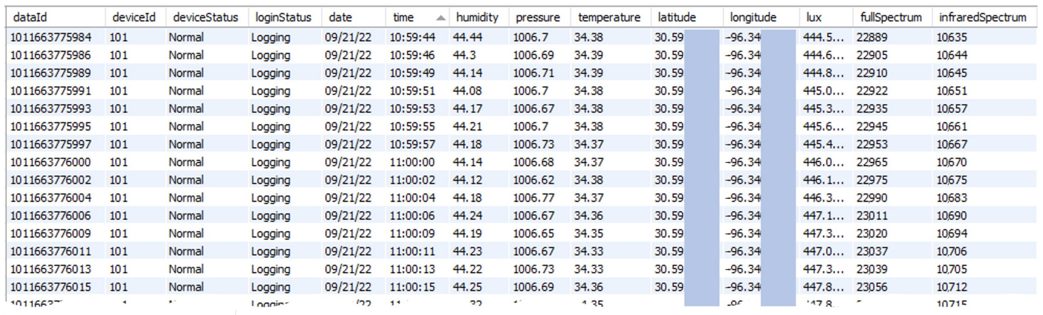

Start of measurement task
Remote data station collecting microclimate data
GPS, humidity, temperature, pressure, lux, full spectrum, and infrared spectrum data
Data publish from remote data station to cluster's MQTT broker
Cluster's MQTT broker receive the data
Data receive
No
Yes
Data store into the database
MySQL
End of measurement task

**Figure 5.** Flowchart showing the sub-process of microclimate data transferring from a remote data station to the database.



| dataId | deviceId | deviceStatus | loginStatus | date | time | humidity | pressure | temperature | latitude | longitude | lux | fullSpectrum | infraredSpectrum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1011663775984 | 101 | Normal | Logging | 09/21/22 | 10:59:44 | 44.44 | 1006.7 | 34.38 | 30.59[redacted] | -96.34[redacted] | 444.5... | 22889 | 10635 |
| 1011663775986 | 101 | Normal | Logging | 09/21/22 | 10:59:46 | 44.3 | 1006.69 | 34.39 | 30.59[redacted] | -96.34[redacted] | 444.6... | 22905 | 10644 |
| 1011663775989 | 101 | Normal | Logging | 09/21/22 | 10:59:49 | 44.14 | 1006.71 | 34.39 | 30.59[redacted] | -96.34[redacted] | 444.8... | 22910 | 10645 |
| 1011663775991 | 101 | Normal | Logging | 09/21/22 | 10:59:51 | 44.08 | 1006.7 | 34.38 | 30.59[redacted] | -96.34[redacted] | 445.0... | 22922 | 10651 |
| 1011663775993 | 101 | Normal | Logging | 09/21/22 | 10:59:53 | 44.17 | 1006.67 | 34.38 | 30.59[redacted] | -96.34[redacted] | 445.3... | 22935 | 10657 |
| 1011663775995 | 101 | Normal | Logging | 09/21/22 | 10:59:55 | 44.21 | 1006.7 | 34.38 | 30.59[redacted] | -96.34[redacted] | 445.6... | 22945 | 10661 |
| 1011663775997 | 101 | Normal | Logging | 09/21/22 | 10:59:57 | 44.18 | 1006.73 | 34.37 | 30.59[redacted] | -96.34[redacted] | 445.4... | 22953 | 10667 |
| 1011663776000 | 101 | Normal | Logging | 09/21/22 | 11:00:00 | 44.14 | 1006.68 | 34.37 | 30.59[redacted] | -96.34[redacted] | 446.0... | 22965 | 10670 |
| 1011663776002 | 101 | Normal | Logging | 09/21/22 | 11:00:02 | 44.12 | 1006.62 | 34.38 | 30.59[redacted] | -96.34[redacted] | 446.1... | 22975 | 10675 |
| 1011663776004 | 101 | Normal | Logging | 09/21/22 | 11:00:04 | 44.18 | 1006.77 | 34.37 | 30.59[redacted] | -96.34[redacted] | 446.3... | 22990 | 10683 |
| 1011663776006 | 101 | Normal | Logging | 09/21/22 | 11:00:06 | 44.24 | 1006.67 | 34.36 | 30.59[redacted] | -96.34[redacted] | 447.1... | 23011 | 10690 |
| 1011663776009 | 101 | Normal | Logging | 09/21/22 | 11:00:09 | 44.19 | 1006.65 | 34.35 | 30.59[redacted] | -96.34[redacted] | 447.3... | 23020 | 10694 |
| 1011663776011 | 101 | Normal | Logging | 09/21/22 | 11:00:11 | 44.23 | 1006.67 | 34.33 | 30.59[redacted] | -96.34[redacted] | 447.0... | 23037 | 10706 |
| 1011663776013 | 101 | Normal | Logging | 09/21/22 | 11:00:13 | 44.22 | 1006.73 | 34.33 | 30.59[redacted] | -96.34[redacted] | 447.3... | 23039 | 10705 |
| 1011663776015 | 101 | Normal | Logging | 09/21/22 | 11:00:15 | 44.25 | 1006.69 | 34.36 | 30.59[redacted] | -96.34[redacted] | 447.8... | 23056 | 10712 |

**Figure 6.** Various data from the device ID 101 database.

The flowchart in Figure 7 shows the data access process using an Android application or web browser. Using an API (Application Programming Interface) URL (Uniform Resource Locator) from the Flash API, the Android application or web browser can use an HTTP (Hypertext Transfer Protocol) GET request to receive a response from the cluster. Once the Flask API receives the request, the program can fetch microclimate data from the MySQL database. The microclimate data can be formatted into JSON (JavaScript Object Notation) form and transferred to the Android application or web browser. Once the Android application or web browser receives the data, it will decode the JSON and display each parameter's value.

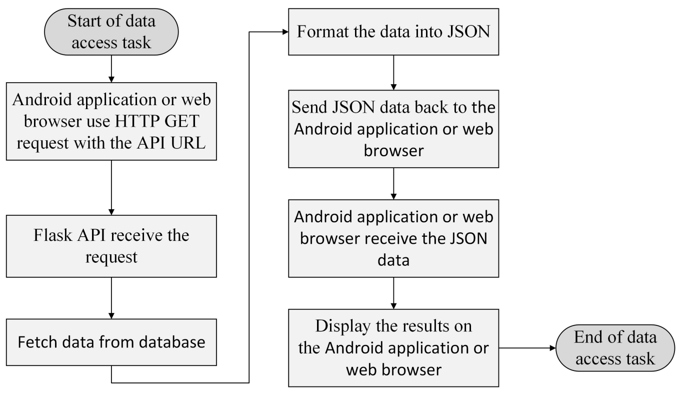
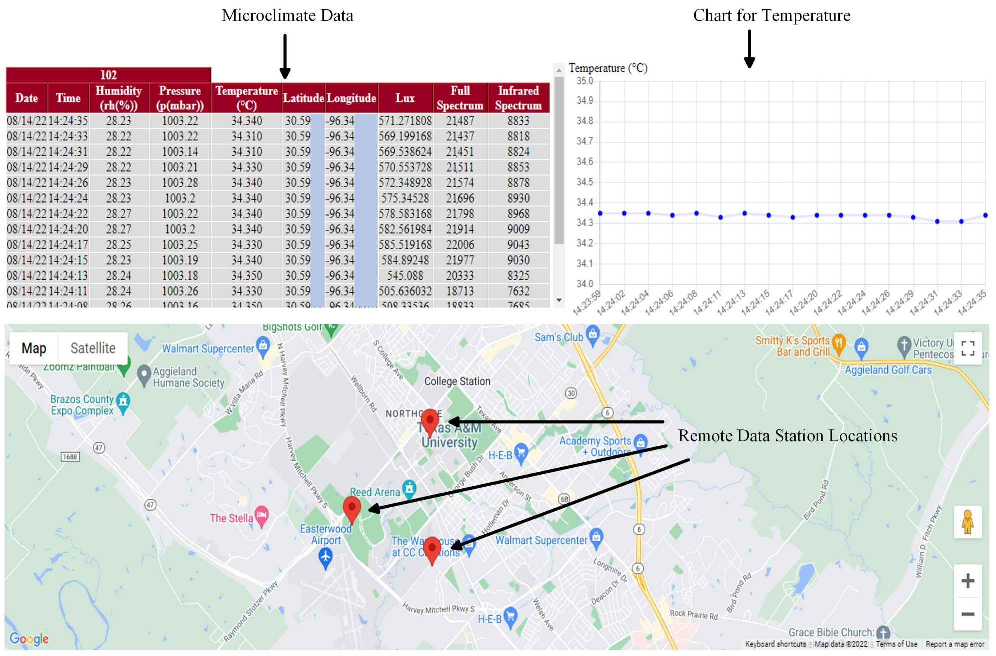

Start of data access task
Android application or web browser use HTTP GET request with the API URL
Flask API receive the request
Fetch data from database
Format the data into JSON
Send JSON data back to the Android application or web browser
Android application or web browser receive the JSON data
Display the results on the Android application or web browser
End of data access task

**Figure 7.** Flowchart showing the data access via Android application and web browser using the Flask API.

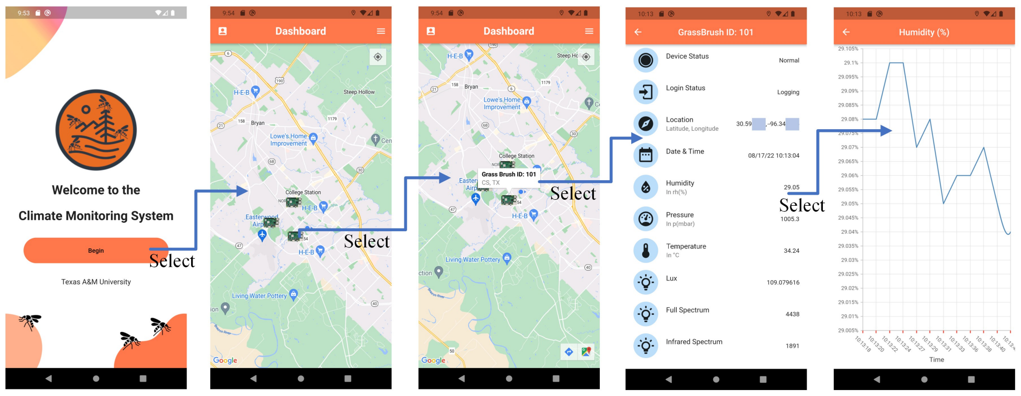
Figure 8 shows screenshots of the website when the data are accessed. The top left table shows the list of available microclimate data. The real-time data graph for that corresponding subtitle is displayed on the top right side by selecting an item from the table. For example, selecting the temperature item can display its real-time data graph. Lastly, the bottom map shows the remote data station locations.



| 102 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Date | Time | Humidity (rh(%)) | Pressure (p(mbar)) | Temperature (°C) | Latitude | Longitude | Lux | Full Spectrum | Infrared Spectrum |
| 08/14/22 | 14:24:35 | 28.23 | 1003.22 | 34.340 | 30.59 | -96.34 | 571.271808 | 21487 | 8833 |
| 08/14/22 | 14:24:33 | 28.22 | 1003.22 | 34.310 | 30.59 | -96.34 | 569.199168 | 21437 | 8818 |
| 08/14/22 | 14:24:31 | 28.22 | 1003.14 | 34.310 | 30.59 | -96.34 | 569.538624 | 21451 | 8824 |
| 08/14/22 | 14:24:29 | 28.22 | 1003.21 | 34.330 | 30.59 | -96.34 | 570.553728 | 21511 | 8853 |
| 08/14/22 | 14:24:26 | 28.23 | 1003.28 | 34.340 | 30.59 | -96.34 | 572.348928 | 21574 | 8878 |
| 08/14/22 | 14:24:24 | 28.23 | 1003.2 | 34.340 | 30.59 | -96.34 | 575.34528 | 21696 | 8930 |
| 08/14/22 | 14:24:22 | 28.27 | 1003.22 | 34.340 | 30.59 | -96.34 | 578.583168 | 21798 | 8968 |
| 08/14/22 | 14:24:20 | 28.27 | 1003.2 | 34.340 | 30.59 | -96.34 | 582.561984 | 21914 | 9009 |
| 08/14/22 | 14:24:17 | 28.25 | 1003.25 | 34.330 | 30.59 | -96.34 | 585.519168 | 22006 | 9043 |
| 08/14/22 | 14:24:15 | 28.23 | 1003.19 | 34.340 | 30.59 | -96.34 | 584.89248 | 21977 | 9030 |
| 08/14/22 | 14:24:13 | 28.24 | 1003.18 | 34.350 | 30.59 | -96.34 | 545.088 | 20333 | 8325 |
| 08/14/22 | 14:24:11 | 28.24 | 1003.26 | 34.330 | 30.59 | -96.34 | 505.636032 | 18713 | 7632 |
| 08/14/22 | 14:24:08 | 28.26 | 1003.16 | 34.350 | 30.59 | -96.34 | 508.33536 | 18833 | 7685 |

**Figure 8.** Data access over the website.

Figure 9 shows screenshots of an Android application developed using Flutter and Android Studio. We can access the same data as the website with a few selections. The first screenshot on the left shows the welcome page. After researchers select the begin button, the second page will show up, and the locations of the remote data stations can be displayed on the map. This map is obtained using Google Map API. Next, users can select icons of remote stations. After that, a message window will pop up. The users can choose the message window, and the application will direct them to the next page, where the list of the available microclimate data is displayed on the screen. Lastly, the users can select the parameter and display the corresponding real-time data graph. For example, selecting the humidity parameter can display its real-time data graph. The users can return to the previous page by selecting the top left return icon.



GrassBrush ID: 101

| | |
|---|---|
| Device Status | Normal |
| Login Status | Logging |
| Location<br>Latitude, Longitude | 30.59[redacted],-96.34[redacted] |
| Date & Time | 08/17/22 10:13:04 |
| Humidity<br>In rh(%) | 29.05 |
| Pressure<br>In p(mbar) | 1005.3 |
| Temperature<br>In °C | 34.24 |
| Lux | 109.079616 |
| Full Spectrum | 4438 |
| Infrared Spectrum | 1891 |

**Figure 9.** Data access over the Android application.

## 3. Measurements

### *3.1. Microclimate Data Measurements*

The microclimate measurements are collected from different remote data stations located in different locations. As shown in Figure 10, the measurements' date is 28 August 2022, and it shows the time duration of 12 h, from 9:30 a.m. to 9:30 p.m. Two sets of data from remote data stations are selected and presented as an example for comparison. Therefore, the set of measurements of (a), (b), and (c) shows the data from the remote data station located in College Station, Texas, USA. Parts of the measurement are missing because the remote station was out of battery at that time. The other set of data in (a), (b), and (c) in Figure 10 is the publicly available weather data collected from Weather Underground (WU). The WU data points that are closest to the test location were selected. The distance between the remote data station and the selected WU station is separated by about 1.6 miles. The light information is not available from Weather Underground, but it was measured using remote data stations as shown in (c) and (f) in Figure 10.

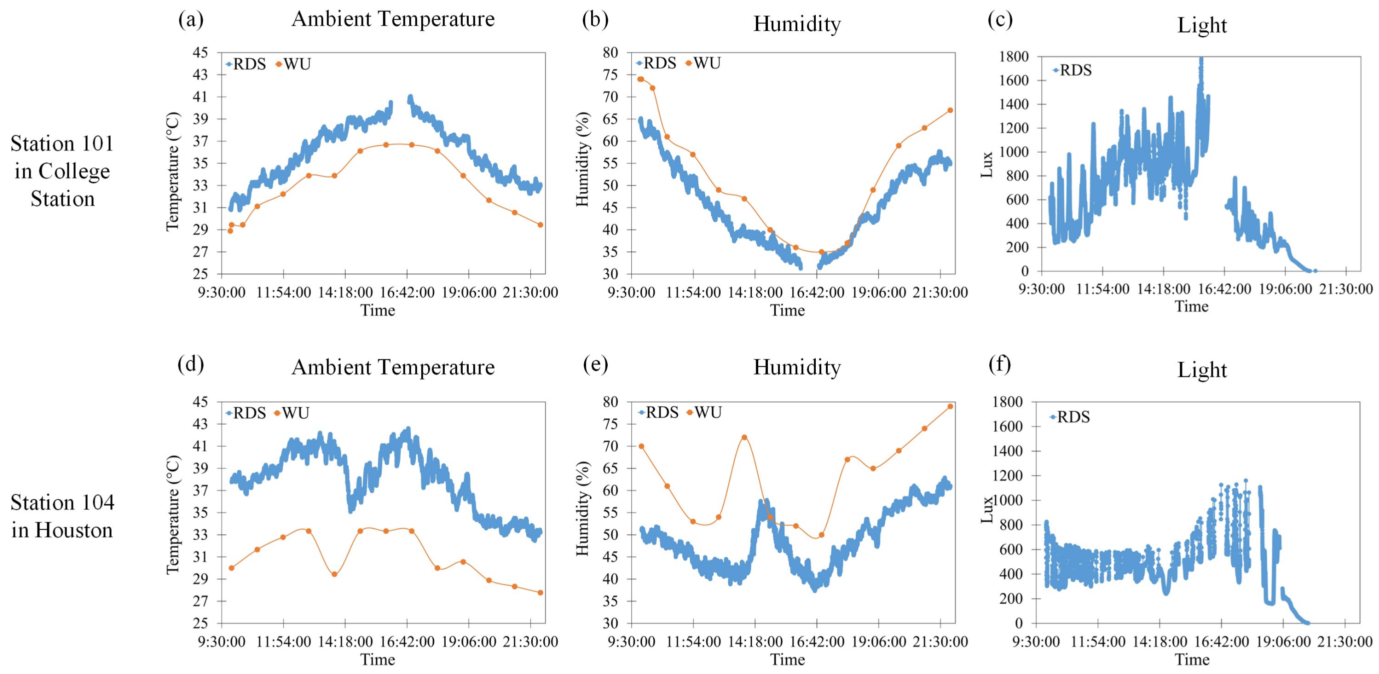


**Figure 10.** Data measured on 28 August 2022 using remote data stations and the data from Weather Underground for comparison. Graphs (**a**–**c**) are the set for the test spot in College Station. And graphs (**d**–**f**) are the set for the test spot in Houston.

The set of measurements of (d), (e), and (f) shows the data from the Remote Data Station (RDS) located in Houston, Texas, USA. In this case, the distance between the RDS and the selected WU station is separated by about 15.8 miles. Data collected using the remote data station are drawn in blue color, and the data from Weather Underground are illustrated in an orange color [28,29].

Users can compare the ambient temperature and humidity data between the ones from the Remote Data Station (RDS) and the ones from Weather Underground (WU). The percentage error between the measurements from the RDS and WU is shown in Table 1. Additionally, it was calculated by using the formula below. The observed value is the data from RDS. The WU value is the data from Weather Underground.

$$\%\ Error = 100 * |Observed\ Value - WU\ Value| \div WU\ Value$$

**Table 1.** Percentage error between the measurements from remote data station and the measurements from Weather Underground.

| Microclimate | Average Percentage Error | Maximum Percentage Error | Minimum Percentage Error |
|---|---|---|---|
| College Station's Ambient Temperature | 5.32% | 7.67% | 2.70% |
| Houston's Ambient Temperature | 14.28% | 24.77% | 5.49% |
| College Station's Humidity | 11.48% | 17.86% | 0.07% |
| Houston's Humidity | 21.69% | 43.74% | 0.12% |

In Table 1, excluding the table's subtitle, the first and second rows show the percentage error for the corresponding chart in row (a) of Figure 10. The third and fourth row show the percentage error for the corresponding graph in row (b) of Figure 10. In College Station, the RDS and WU station are close to each other, so the various percentage errors for ambient temperature and humidity are smaller than the ones from Houston. However, in Houston, the RDS is far from the WU station, so the percentage error is higher than those from College Station. The average percentage error for College Station's temperature is 5.32%, which is equivalent to the minimum percentage error for Houston's temperature. Additionally, in temperature, the error caused by the device can be considered as ±2.50 °C. This error factor would not be regarded as a significant contributing factor that shows the difference between the data from the RDS and WU. These results indicate that this proposed microclimate IoT system may provide more accurate data for the test spots for selected potential breeding locations. Additionally, more accurate and precise data collection significantly changes research quality in that area.

Instead of showing microclimate data for 12 h, Table 2 shows the extended period of the daily record for two of the recorded variables, ambient temperature and humidity. These data were collected from another station, 102, in College Station. Therefore, the table is divided into two big columns, College Station's station 102 and WU. Under the College Station column, the average and standard deviation of ambient temperature and humidity are shown from 7 September to 14 September 2022. Because WU does not provide the standard deviation, only the average temperature and humidity are shown under the second big column, WU. The distance between station 102 and the selected data point from WU is separated by about 1.14 miles.

On September 7th, the average temperature for station 102 in College Station was 27.57 °C, and the average temperature for the nearby Weather Underground station was 26.94 °C. The percentage error between those two data points is 2.34%, close to the minimum percentage error, 2.70%, for College Station's ambient temperature in Table 1. Therefore, mosquito researchers could use these accurate microclimate data for the area of interest to generate more accurate analysis. In the following subsection, additional ping measurements for the cluster are described for more information.

**Table 2.** Daily record of temperature and humidity for the dates between 7 September and 14 September 2022.

| Date | Station 102 in College Station | | | | Weather Underground | |
|---|---|---|---|---|---|---|
| | Temperature (°C) | | Humidity (%) | | Temperature (°C) | Humidity (%) |
| | Average | Std. Dev. | Average | Std. Dev. | Average | Average |
| 9/7/22 | 27.57 | 3.90 | 70.90 | 15.36 | 26.94 | 75.70 |
| 9/8/22 | 28.32 | 3.52 | 58.94 | 14.76 | 27.53 | 64.60 |
| 9/9/22 | 28.43 | 1.49 | 58.10 | 4.72 | 27.50 | 61.50 |
| 9/10/22 | 29.91 | 2.90 | 54.62 | 14.62 | 28.06 | 61.30 |
| 9/11/22 | 28.97 | 3.02 | 64.32 | 15.53 | 27.94 | 62.30 |
| 9/12/22 | 27.87 | 3.50 | 50.42 | 11.16 | 26.78 | 55.80 |
| 9/13/22 | 27.49 | 4.48 | 53.47 | 7.59 | 26.50 | 58.00 |
| 9/14/22 | 27.23 | 4.63 | 65.61 | 8.20 | 26.33 | 55.30 |

### *3.2. Ping Measurements between the Master Node and the Worker Nodes in the Cluster*

This section started by comparing ping time between the master node and worker nodes. It is under the scenario of four remote data stations simultaneously sending microclimate data to their connected cluster. In this measurement, a 32 bytes data packet is used to measure the ping time in millisecond (ms) units. Table 3 shows the ping time between the master and worker nodes for the cluster. The table shows the measurements' the minimum time, average time, maximum time, and standard deviation.

**Table 3.** The ping time between rpi4b-master-1 and each of its workers.

| Master Node | Worker Node | Ping Time | | | |
|---|---|---|---|---|---|
| | | Minimum Time (ms) | Average Time (ms) | Maximum Time (ms) | Standard Deviation (ms) |
| rpi4b-master-1 | rpi3b-plus-worker-1 | 0.614 | 0.759 | 0.913 | 0.067 |
| | rpi3b-plus-worker-2 | 0.598 | 0.772 | 0.982 | 0.081 |
| | rpi3b-plus-worker-3 | 0.545 | 0.754 | 0.933 | 0.076 |

Table 3 is divided into three big columns. The first column is the master node of the cluster from Figure 2. The second column is the worker nodes, rpi3b-plus-worker-1, rpi3b-plus-worker-2, and rpi3b-plus-worker-3, from Figure 2. The last column is the ping time, which is divided into four sub-columns, minimum time, average time, maximum time, and standard deviation. The average time between the RPi 4 B master node 1 and the RPi 3 B+ worker node 1 is 0.759 ms. The standard deviation between them is 0.067 ms. By using the average time and standard deviation, the standard deviation graph can be made for a graphical representation of the data.

## 4. Conclusions

In this paper, we proposed a microclimate IoT system for mosquito research applications, and it may collect and monitor accurate microclimate data from selected locations. Mosquito researchers may use the microclimate data for the test spots of interest for analysis and control. Various methods of accessing the data in this proposed IoT system are offered in this IoT system. The data can be accessed using an Android application and website for the data stored in the Raspberry Pi Micro-Kubernetes cluster. The measurements were shown, and the data were compared with the publicly available weather data. The result shows that this customized IoT network, remote data stations, and low-cost RPi MicroK8s cluster may provide more accurate data for the test spots, and the data can be accessed via an android application and website.

For future work, we will continue to develop various RPi MicroK8s clusters. We plan to perform investigations using the microclimate data from this IoT system. We plan to check if there is a substantial limitation on the accuracy and precision of micro-climate measurements. Additionally, we plan to carry out more granular monitoring for micro-climates to check the difference between the micro-climate and macro-climate in mosquito breeding sites. There can be multiple opportunities for future work that can build upon the proposed cluster in this study. More different types of measurements at multiple locations can be performed and processed using the proposed clusters. We plan to improve the cluster by using a newer version of RPi for all nodes in the cluster. We plan to perform more experimentation by increasing the scale of worker nodes by adding several more Raspberry Pis to the cluster. In the remote data station of the IoT system architecture, we see a potential vision of making the work easier for the maintenance engineer by making wireless sensor end-node devices need less maintenance and minimizing their energy use.

**Author Contributions:** Conceptualization, Z.P., B.H., K.M. and Z.A.; methodology, Z.P., B.H., K.M. and Z.A.; software, Z.P.; validation, Z.P.; formal analysis, Z.P. and B.H.; investigation, Z.P. and B.H.; resources, Z.P., B.H., K.M. and Z.A.; data curation, Z.P.; writing—original draft preparation, Z.P. and B.H.; writing—review and editing, Z.P. and B.H.; supervision, B.H., K.M. and Z.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported through Cooperative Agreement Number 1U01CK000512 between the Centers for Disease Control and Prevention (CDC) and the University of Texas Medical Branch/Western Gulf Center for Excellence in Vector-Borne Diseases. Moreover, this work was supported by Dr. B. Hur's Texas A&M research fund and resources.

**Data Availability Statement:** Data for a few specific days is contained within the article. The data for other days are not publicly available due to the funding source's ownership of the data.

**Acknowledgments:** We would like to thank Eun Jung Kim and Ana Goulart for their valuable insights and support. We would like to thank Kevin Price, Fernando Aguado, and Jonathan Gavlick for their help during the early stages of this research.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| API | Application Programming Interfaces |
| CDC | Centers for Disease Control and Prevention |
| CPU | Central Processing Unit |
| GPS | Global Positioning System |
| HTTP | Hypertext Transfer Protocol |
| ID | Light-Emitting Diode |
| IoT | Internet of Things |
| JSON | JavaScript Object Notation |
| K8s | Kubernetes |
| LED | Light-Emitting Diode |
| LTS | Long Term Support |
| MicroK8s | Micro-Kubernetes |
| MQTT | MQ Telemetry Transport |
| PCB | Printed Circuit Board |
| RDS | Remote Data Station |

| | |
|---|---|
| RGB | Red, Green, and Blue |
| RPi | Raspberry Pi |
| Std. Dev. | Standard Deviation |
| URL | Uniform Resource Locator |
| WU | Weather Underground |

## References

1. Ronca, S.E.; Ruff, J.C.; Murray, K.O. A 20-year historical review of West Nile virus since its initial emergence in North America: Has West Nile virus become a neglected tropical disease. *PLoS Neglected Trop. Dis.* **2021**, *15*, e0009190. [CrossRef] [PubMed]
2. Gatton, M.; Chitnis, N.; Churcher, T.; Donnelly, M.; Ghani, A.; Godfray, C.; Gould, F.; Hastings, I.; Marshall, J.; Ranson, H.; et al. The Important of Mosquito Behavioral adaptations to Malaria Control in Africa. *Int. J. Org. Evol.* **2013**, *67*, 1218–1330. [CrossRef] [PubMed]
3. Roundy, C.M.; Azar, S.R.; Rossi, S.L.; Huang, J.H.; Leal, G.; Yun, R.; Fernandez-Salas, I.; Vitek, C.J.; Paploski, I.A.; Kitron, U.; et al. Variation in Aedes aegypti Mosquito Competence for Zika Virus Transmission. *Emerg Infect Dis.* **2017**, *23*, 625–632. [CrossRef] [PubMed]
4. Bai, L.; Morton, C.; Liu, Q. Climate change and mosquito-borne diseases in China: A review. *Glob. Health* **2013**, *9*, 10. [CrossRef] [PubMed]
5. Ryan, S.J.; Carlson, C.J.; Mordecai, E.A.; Johnson, L.R. Global expansion and redistribution of Aedes-borne virus transmission risk with climate change. *PLoS Neglected Trop. Dis.* **2019**, *13*, e0007213. [CrossRef] [PubMed]
6. Pley, C.; Evans, M.; Lowe, R.; Montgomery, H.; Yacoub, S. Digital and technological innovation in vector-borne disease surveillance to predict, detect, and control climate-driven outbreaks. *Lancet Planet. Health* **2021**, *5*, e739–e745. [CrossRef] [PubMed]
7. Davis, J.K.; Vincent, G.; Hildreth, M.B.; Kightlinger, L.; Carlson, C.; Wimberly, M.C. Integrating Environmental Monitoring and Mosquito Surveillance to Predict Vector-borne Disease: Prospective Forecasts of a West Nile Virus Outbreak. *PLoS Curr.* **2017**, *9*, 2157–3999. [CrossRef] [PubMed]
8. Balatamoghna, B.; Jaganath, A.; Vaideeshwaran, S.; Subramanian, A.; Suganthi, K. Integrated balancing approach for hosting services with optimal efficiency - Self Hosting with Docker. *Mater. Today Proc.* **2022**, *62*, 4612–4619. [CrossRef]
9. Shah, J.; Dubaria, D. Building Modern Clouds: Using Docker, Kubernetes Google Cloud Platform. In Proceedings of the 2019 IEEE 9th Annual 261 Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 0184–0189.
10. Containers, VMs, Kubernetes and VMware. Available online: http://googlecloudplatform.blogspot.com/2014/08/containers-vms-kubernetes-and-vmware.html (accessed on 21 September 2022).
11. Micheli, P.; Johnson, M.; Godsell, J. Editorial How the Covid-19 pandemic has affected, and will affect, operations and supply chain management research and practice. *Emerald Publ. Ltd.* **2021**, *41*, 773–780. [CrossRef]
12. Nakamura, K.; Manzoni, P.; Redondi, A.; Longo, E.; Zennaro, M.; Cano, J.; Calafate, C. A LoRa-based protocol for connecting IoT edge computing nodes to provide small-data-based services. *Digit. Commun. Netw.* **2022**, *8*, 257–266. [CrossRef]
13. Khazaei, H.; Bannazadeh, H.; Leon-Garcia, A. SAVI-IoT: A Self-Managing Containerized IoT Platform. In Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, 21–23 August 2017 pp. 227–234.
14. Hur, B.; Myles, K.; Adelman, Z.; Erraguntla, M.; Lawley, M.; Kim, E.; Burgi, J.; Price, K.; Fritz, K.; Stalcup, D.; et al. IoT Environmental-monitoring System Development for Mosquito Research Through Capstone Project Integration in Engineering Technology. In Proceedings of the 2021 ASEE Virtual Annual Conference Content Access, Virtual, 26–29 July 2021.
15. Hur, B.; Myles, K.; Adelman, Z.; Kim, S.; Kim, E.J.; Price, K.; Gavlick, J. Low-Cost Raspberry Pi Compute Module 3+ Cluster for Mosquito Research via Capstone Project. In Proceedings of the 2022 ASEE Annual Conference & Exposition, Minneapolis, MN, USA, 26–29 June 2022.
16. Paaijmans, K.P.; Thomas, M.B. The influence of mosquito resting behaviour and associated microclimate for malaria risk. *Malar. J.* **2011**, *10*, 183. [CrossRef] [PubMed]
17. Murdock, C.C.; Evans, M.V.; McClanahan, T.D.; Miazgowicz, K.L.; Tesla, B. Fine-scale variation in microclimate across an urban landscape shapes variation in mosquito population dynamics and the potential of Aedes albopictus to transmit arboviral disease. *PLoS Neglected Trop. Dis.* **2017**, *11*, e0005640. [CrossRef] [PubMed]
18. Leccese, F.; Cagnetti, M.; Trinca, D. A Smart City Application: A Fully Controlled Street Lighting Isle Based on Raspberry-Pi Card, a ZigBee Sensor Network and WiMAX. *MDPI Sens.* **2014**, *14*, 24408–24424. [CrossRef] [PubMed]
19. Ambrož, M. Raspberry Pi as a low-cost data acquisition system for human powered vehicles. *Measurement* **2017**, *100*, 7–18. [CrossRef]
20. Scuro, C.; Sciammarella, P.F.; Lamonaca, F.; Olivito, R.S.; Carni, D.L. IoT for structural health monitoring. *IEEE Instrum. Meas. Mag.* **2018**, *21*, 4–14. [CrossRef]
21. Polineni, S.; Shastri, O.; Bagchi, A.; Gnanakumar, G.; Rasamsetti, S.; Sundaravadivel, P. MOSQUITO EDGE: An Edge-Intelligent Real-Time Mosquito Threat Prediction Using an IoT-Enabled Hardware System. *MDPI Sens.* **2022**, *22*, 695. [CrossRef] [PubMed]

22. Javed, A. Container-Based IoT Sensor Node on Raspberry Pi and the Kubernetes Cluster Framework. Master's Thesis, Aalto University, Espoo, Finland, 24 August 2016.
23. MicroK8s Documentation-Home. Available online: https://microk8s.io/docs (accessed on 1 October 2022).
24. Kang, B.; Jeong, J.; Choo, H. Docker Swarm and Kubernetes Containers for Smart Home Gateway. *IT Prof.* **2021**, *23*, 75–80. [CrossRef]
25. Debauche, O.; Mahmoudi, S.; Guttadauria, A. A New Edge Computing Architecture for IoT and Multimedia Data Management. *Information* **2022**, *13*, 89. [CrossRef]
26. Rautmare, S. MySQL and NoSQL database comparison for IoT application. In Proceedings of the 2016 IEEE International Conference on Advances in Computer Applications (ICACA), Coimbatore, India, 24 October 2016; pp. 235–238.
27. Tongkaw, S.; Tongkaw, A. A comparison of database performance of MariaDB and MySQL with OLTP workload. In Proceedings of the 2016 IEEE Conference on Open Systems (ICOS), Langkawi, Malaysia, 10–12 October 2016; pp. 117–119.
28. College Station, TX Weather History. Available online: https://www.wunderground.com/history/daily/us/tx/college-station/KCLL/date/2022-8-29 (accessed on 31 August 2022).
29. Houston, TX Weather History. Available online: https://www.wunderground.com/history/daily/us/tx/houston/KHOU/date/2022-8-29 (accessed on 31 August 2022).