21

NAMIBIA UNIVERSITY OF SCIENCE AND TECHNOLOGY
Faculty of Engineering

Department of Mechanical, Industrial and Electrical Engineering

Bachelor of Engineering in Electronics and Telecommunications

Water Monitoring System for Farms

Prepared for: Prof Oyedokun Zacchaeus

Prepared by: Ester Puye-Ipawa Ndatyoonawa Ndadi
220106428

24 May 2024

Acknowledgement

Firstly, the author is deeply grateful and thankful to Christ, for it is only through Him that anything is possible, and to the University of Science and Technology, especially the Department of Mechanical, Industrial, and Electrical Engineering. This course's debut and the learning opportunities it offers have sparked an innovative and creative attitude.

This module would not have been finished without mentor Dr. Oyedokun's crucial assistance and support. His expert and fervent guidance enhanced the design, guaranteeing that the project was completed on schedule and in compliance with the course requirements.

The author as well wishes to acknowledge the assistance received for the compilation of this guide from the notes and publications of:

- (Logan Vandermark, 2023)
- (Nalina Suresh, 2019)
- (Yash Panchal, 2023)

Lastly, a sincere thank you to family and friends whose inspiration and support through trying times gave me the willpower to go on. A special thank you goes out to everyone who offered helpful advice and criticism; without you this journey would not have been possible.

List of Acronyms

- IoT – Internet of Things
- GPIO – General Purpose Input/Output
- IP – Internet Protocol
- LED – Light Emitting Diode
- SG Servo – Standard Gear
- DC – Direct Current
- HTML – Hypertext Markup Language
- AWS – Amazon Web Services

Terms of reference

The IoT-Automated Workspace project coordinator and lecturer, Dr. Oyedokun, has asked that a final report be completed. An extensive technical report must be included with every project. To make it easy for other technicians to duplicate the project, this report should include comprehensive, step-by-step instructions on how the project was carried out. A detailed summary of all important needs and specifications should be provided.

Abstract

This project describes the design, development, and testing of a Water Monitoring System for Farms. This system utilizes Internet of Things (IoT) technology to ensure animals have consistent access to fresh water while optimizing water resource management on farms. The core functionalities include real-time water level monitoring through a water level sensor, automated refilling with a pump controlled by a Raspberry Pi microcontroller unit, and IP address-based control within the farm network. The project successfully tackled challenges like merging component codes and addressed power limitations by incorporating a separate power supply and relay.

Future enhancements could involve integrating additional sensors for water quality, temperature, or animal activity monitoring. Cloud-based monitoring on a Global platform could be implemented to enhance security, scalability, and enable remote control from anywhere around the world. Data analytics and machine learning could be utilized to identify trends in water usage and animal behavior, potentially leading to optimized refilling schedules and resource allocation. By offering real-time data, automation, and the potential for remote access and advanced analytics, this Water Monitoring System for Farms has the potential to significantly improve farm management practices, promote animal welfare, and contribute to a more sustainable and efficient agricultural industry.

Table of Contents

Introduction

In many areas around Namibia, people rely more on farming in terms of occupation and people’s income depend on it. With advancement in technology and ever-changing weather conditions, accurate and affordable water level measurement systems has become necessary for farmers, (Nalina Suresh, 2019). In today's agricultural landscape, ensuring the well-being of farm animals is paramount. This not only aligns with ethical treatment practices but also directly impacts farm productivity. Two critical aspects of animal care are maintaining adequate water levels for hydration and providing essential supplements like salt licks. Traditional methods for addressing these needs often involve manual processes, which can be time-consuming, labor-intensive, and prone to human error. This can lead to potential issues like dehydration in livestock and inefficiencies in resource utilization.

In a move to revolutionize farm management practices, this project introduces a novel solution: the Water Monitoring System for Farms. This innovative system leverages the power of Internet of Things (IoT) technology. It provides real-time monitoring and management of water levels in animal troughs, coupled with the timely provision of salt licks. By integrating advanced sensor technologies with automated control systems and intuitive user interfaces, the system offers a multi-pronged approach to improving farm operations. Firstly, enhanced animal welfare is achieved through real-time data on water levels, ensuring prompt refilling and preventing dehydration. Additionally, the system facilitates the timely provision of salt licks, promoting overall animal health and well-being. Secondly, optimized resource utilization is realized through automated refilling based on actual water levels, minimizing water waste. The system also optimizes energy consumption by eliminating unnecessary pump operation. Thirdly, improved farm productivity is achieved by reducing manual labor associated with water level monitoring and refilling, translating to lower labor costs. Furthermore, healthier animals contribute to improved weight gain, milk production, and reproduction rates, ultimately leading to increased farm productivity. Finally, the Water Monitoring System for Farms promotes sustainability by encouraging responsible water management practices, contributing to a more sustainable agricultural industry.

Key objectives

- **Real-Time Water Level Monitoring and Automated Refilling**: Implementing a water level sensor in the animal’s water trough for continuous monitoring of water and data transmission of real-time data to the Raspberry Pi which is the central hub. Developing an automated refilling system triggered by low water levels to ensure consistent hydration and minimize waste.
- **Animal-Activated Salt Licks:** Integrating an ultrasonic sensor to detect animal presence near troughs and designing a dispensing mechanism using a servo motor controlled by this sensor to provide salt licks only when animals are present.

- **Remote Monitoring and Control:** Developing a user-friendly mobile app for farm managers to remotely monitor water levels, receive alerts and control refilling of water troughs

Literature Review

Author 1:

According to an article written by (Logan Vandermark, 2023), An essential daily task of livestock producers is ensuring that animals have access to water. This often requires frequent checking of water tanks to make sure they are free of ice and filling properly. Depending on the location of animals relative to the producer's home, checking water tanks can require hours of labor and significant fuel costs for remote pastures. Research has indicated that livestock producers may spend over $1,500 a year on fuel checking water sources. Numerous issues can occur that prevent animals from accessing water, including equipment failure and tanks freezing over. The consequences of livestock not having access to water can be severe, ranging from animals escaping from pasture boundaries in search of a water source to declines in animal performance and death, potentially costing tens of thousands of dollars. Checking stock tanks, tire tanks, or stock dams in person on a daily basis is not always an option. Through the use of water monitoring systems and other pieces of technology, you can check the status of the water source remotely throughout the day.

According to (Logan Vandermark, 2023), Cellular based trail cameras can serve as a cost-effective option to remotely view your stock tanks. One of the greatest advantages this can serve is providing visual confirmation of available water. These cameras can also be set up to send you a picture at regular schedules, such as during the morning and evening. These are also quick and easy to install and take down, providing greater flexibility than other monitoring systems for water sources. One consideration to make is whether you have strong cellular service reception at the watering location, which is required to transmit images. The big question with implementing any technology on ranch is, "Does this make sense (or cents) on my operation?" This technology depends upon the user to learn how to use it and trust that it is monitoring the water source accurately. In many cases water monitoring systems will provide added value to most operations through either time saved checking water sources or fuel savings. Before purchasing a unit, several factors should be considered to find the right system for your operation, including: 1) cellular connectivity 2) monthly or yearly subscription fees, and 3) the ability to mount units in a location where cattle can't rub or scratch them.

Summary: This project explored the design and development of a Water Monitoring System for Livestock that leverages Internet of Things (IoT) technology using cameras to check the water levels in tanks. While the current design(Water Monitoring system for farms) focuses on core functionalities like real-time water level monitoring and automated refilling, the potential exists for future enhancements that could rival the functionalities discussed in this literature review.

Author 2:

According to (Yash Panchal, 2023), In this paper, we present the design and implementation of a smart dustbin that utilizes an Arduino micro-controller to control a servo motor and an ultrasonic sensor. The purpose of the dustbin is to improve waste management by providing a convenient and efficient way to dispose of garbage. The smart dustbin operates by detecting the presence of an object using the ultrasonic sensor. When an object is detected, the servo motor opens the lid of the dustbin, allowing the user to dispose of their waste. Once the waste is deposited, the servo motor closes the lid, ensuring that the dustbin remains closed and odor-free. The design of the smart dustbin is simple and easy to assemble, making it an ideal solution for households and public spaces. The Arduino micro-controller is used to control the servo motor and the ultrasonic sensor, allowing for precise and accurate detection of objects. The ultrasonic sensor is mounted on the lid of the dustbin, providing a wide detection range. The smart dustbin can be powered using a battery or an external power source, making it flexible and easy to install. The Arduino micro-controller is programmed using the Arduino IDE, making it easy for anyone to modify or update the software to suit their specific needs. Overall, the smart dustbin presented in this paper provides a practical and effective solution to waste management. It is easy to use, easy to install, and can be customized to suit a wide range of applications. With its ability to detect objects using an ultrasonic sensor and control a servo motor, the smart dustbin represents a significant advancement in waste management technology.

Summary: This paper describes the design and implementation of a smart dustbin for improved waste management. The system leverages an Arduino microcontroller to control a servo motor and an ultrasonic sensor. Function: The ultrasonic sensor detects the presence of an object (user approaching), Upon detection, the servo motor opens the lid for waste disposal. Once the object is removed, the servo motor closes the lid, ensuring it remains closed and odor-free.

The core functionalities of the smart dustbin (Arduino, ultrasonic sensor, servo motor) can be adapted for use in a Water Monitoring System for Farms. In this application: The ultrasonic sensor would be positioned near the water trough.Upon detecting an animal's presence (replacing the object in the smart dustbin), the servo motor could be programmed to perform a new action: dispensing a controlled amount of salt lick.This approach offers the potential benefit of automated salt lick distribution based on animal presence, promoting efficient utilization and ensuring animals receive essential minerals.

Methodology

Procedures

1. The initial phase involved defining the system's functionalities and user requirements. This included real-time water level monitoring, automated refilling, animal presence detection, salt lick dispensing, and remote mobile app control.
2. Based on the defined requirements, appropriate components were chosen. This included a water trough and tank, LED to switch on when the pump is turned off, water level sensor for accurate water level measurement in the water trough, ultrasonic sensor for animal presence detection, reliable water pump for refilling water into the water trough from the tank, servo motor to provide the salt licks when activated by the ultrasonic sensor, Raspberry Pi as the central hub, jumper wires for electrical connections between the components and a suitable server platform for data processing and communication.
3. The testing of components was made to ensure proper functionality of all components
4. The block diagram of how the system’s layout or set up would be was chosen
5. The water level sensor was strategically placed within animal troughs, ensuring proper immersion and accurate water level readings, ultrasonic sensor was positioned near the troughs for optimal

animal presence detection, the pump was placed inside the tank for pumping water into the trough and the servo motor was as well positioned near the water trough for animals to easily get the salt licks.

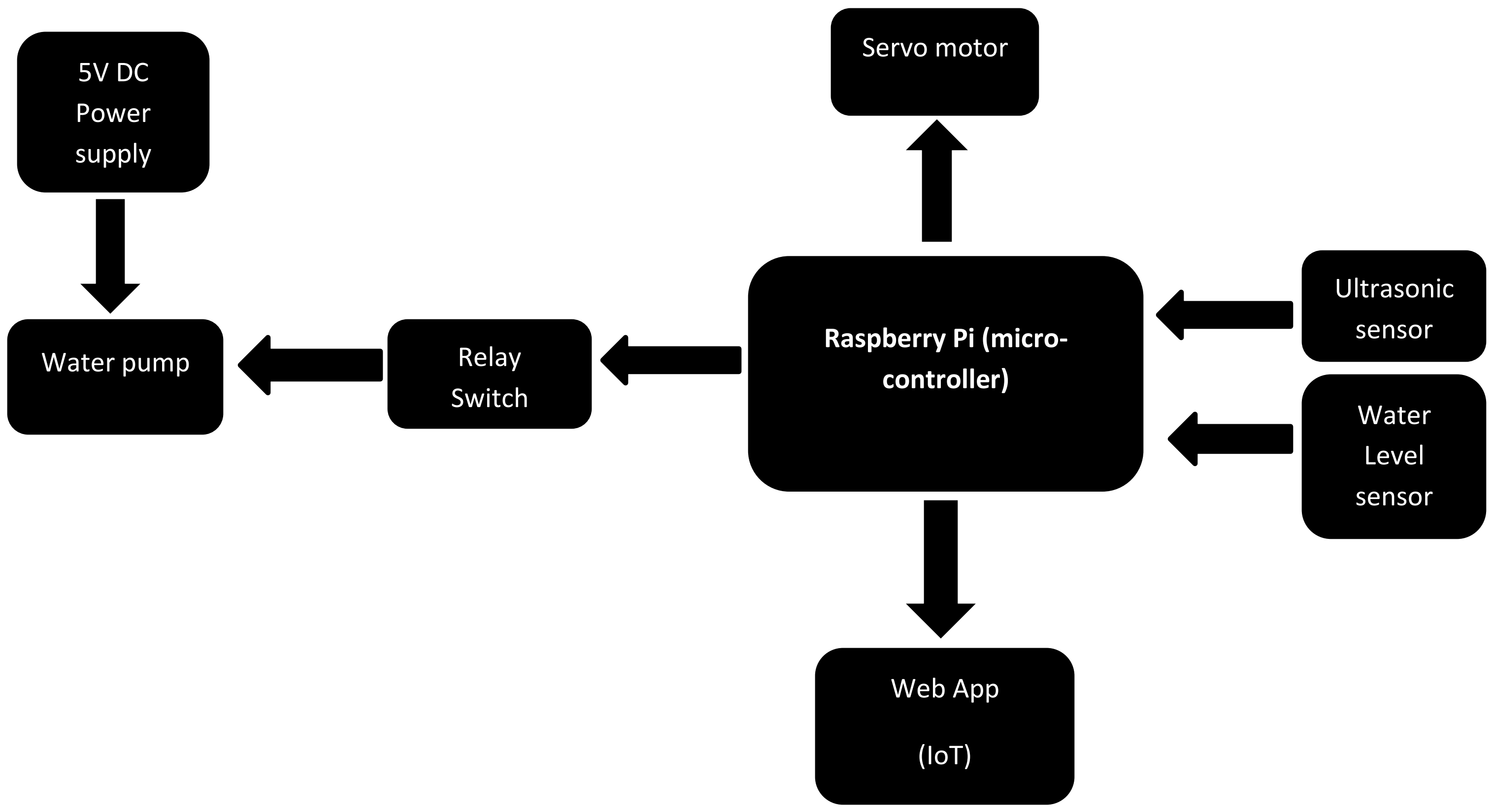
6. A robust data acquisition system was developed to collect and process sensor data. This system involved electrical connections using jumper wires between sensors(water level, ultrasonic), servo motor, pump and the Raspberry Pi for data transmission.
7. A programming logic was developed to trigger the pump based on water level sensor data. When water level sensor reads low, the system would activate the pump to refill the trough automatically and the pump automatically stops pumping when the water level sensor reads high .
8. A programming logic was developed to trigger the servo motor based on the ultrasonic sensor data. When the ultrasonic sensor detects the presence of animals, the system then dispenses the servo motor arm to provide salt licks to animals and when no animal presence is detected by the ultrasonic sensor, then no dispersion of the servo arm.
9. A programming logic was developed for the on and off switching of the LED when the pump is turned off and on respectively
10. The mobile app was programmed to offer functionalities such as real-time water level visualization for the trough, customizable alert notifications for low water levels, animal presence, and system malfunctions. Additionally, the app allows for remote control of the automated water refilling process by the pump
11. Comprehensive testing of the system was conducted to ensure its functionality. This included automated pumping of water into the trough, animal presence detection, salt lick dispensing, and mobile app functionalities.

Software Design

- Fritzing
- Python
- HTML

Hardware Design

Figure 1: The block diagram of the design project



Data Acquisition Unit: Via the Raspberry Pi, the system acquires data from the water level sensor in the farm environment which continuously monitors the water level. This data is then transmitted via a communication module which is Wi-Fi to a cloud server for processing and analysis.

Cloud Server: The cloud server receives water sensor data from the farm environment via Wi-Fi. It then interprets the data to determine water level in the water trough and potential system malfunctions. Based on the processed data, the cloud server sends instructions to the farm controller unit.

Farm Controller Unit: The farm controller unit receives instructions from the cloud server. It controls the system's functionalities through the relay.

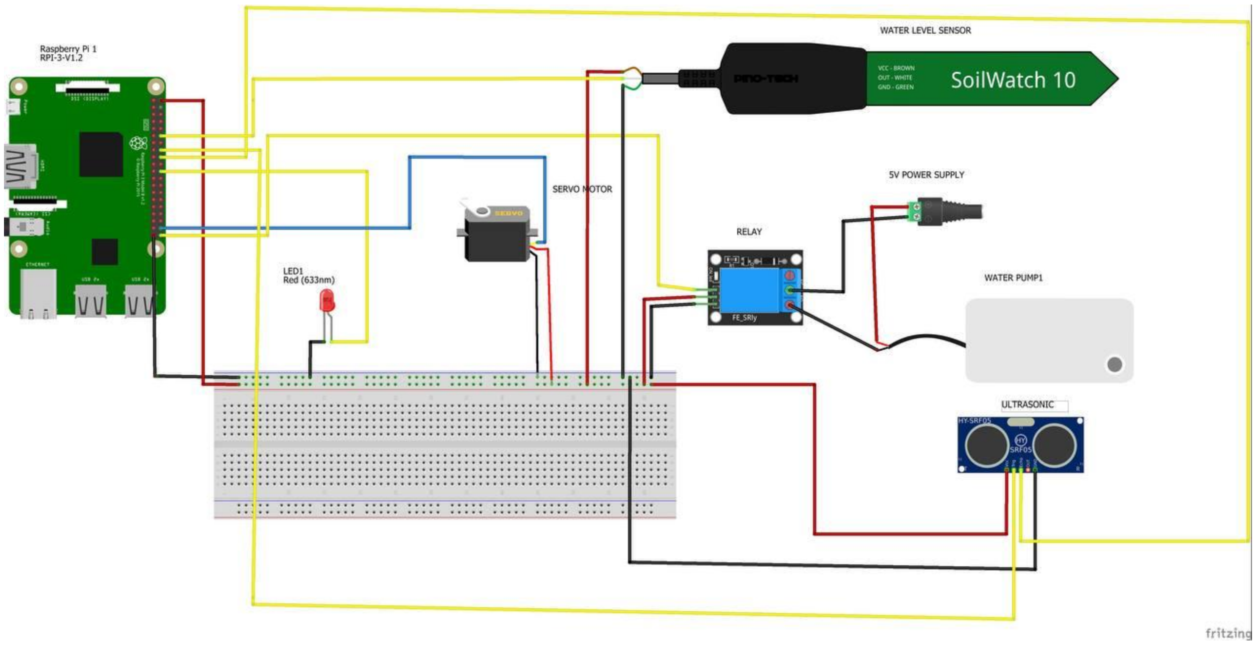
Control and actuation Unit: The control and actuation unit, controlled by the farm controller unit through relays, manages automated processes:

- Water pump: When the cloud server detects low water levels through the water level sensor data, it sends a signal to the farm controller unit, which then activates the pump to refill the trough automatically.
- Salt lick dispenser: The mechanism done by the servo motor for dispensing salt licks is controlled by the ultrasonic sensor. The servo motor dispenses the salt licks to the animals only when the ultrasonic sensor detects the presence of animals near the water trough and it goes back to its original position when the ultrasonic sensor can’t detect anything.

Mobile App (Cloud-based): The system offers a mobile application interface for farm managers to monitor and control the system remotely. The app connects to the cloud server. The mobile app displays: Real-time water level data from the trough water level sensor, Notifications for low water levels , An interface to control the water pump remotely for manual refilling and real-time data from the ultrasonic sensor and the servo motor.

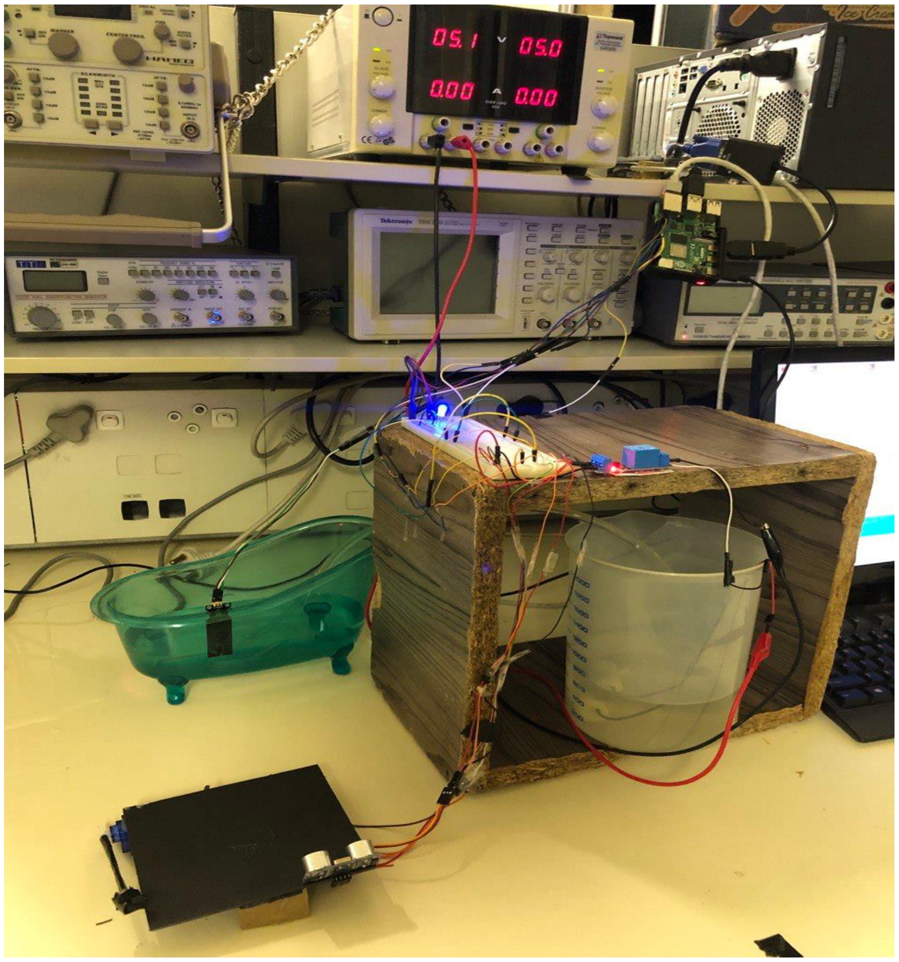
Therefore, the block diagram in Figure 1 showcases a cloud-based water monitoring system for farms, leveraging a central cloud server for data processing and communication. The real-time data acquisition from water level sensors ensures accurate monitoring of water availability. The use of a cloud server allows for remote data access, storage, and analysis, offering greater flexibility and scalability for farm management. The farm controller unit executes commands from the cloud server, controlling actuators like pumps. The mobile application interface empowers farm managers with remote monitoring and control capabilities, improving farm management efficiency.

Figure 2: The circuit in fritzing



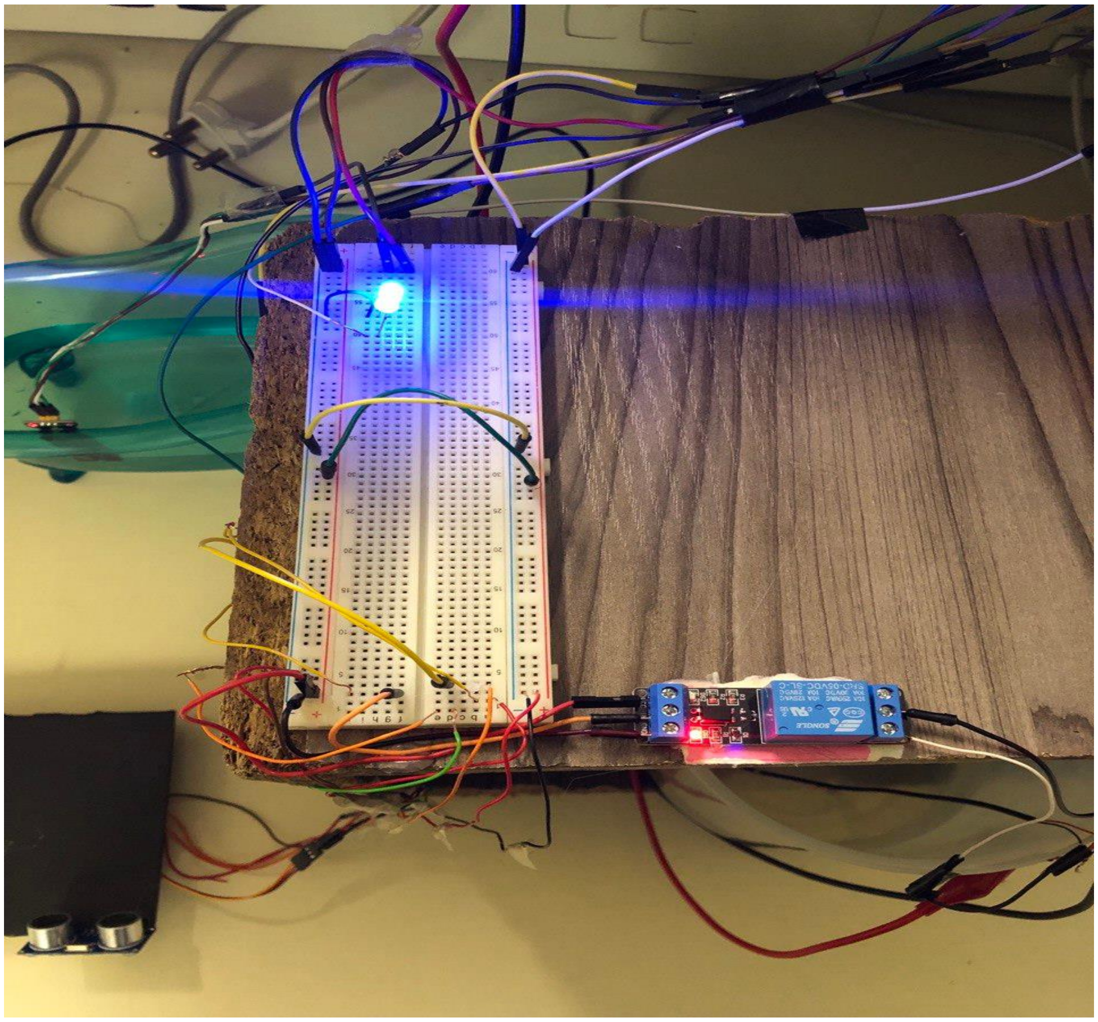
The circuit in Figure 2 is how the whole design was connected. Showing how the components are connected to their signal pins from the Raspberry Pi and how they are connected to the 5V and the ground of the Raspberry Pi, as well how the pump is connected to the 5V DC power supply.

Figure 3: The project design set-up



The set-up in Figure 2 shows all the components used for the Water Monitoring System for farms. A DC power supply is shown in the picture supplying 5.1V to the water pump inside the tank (jar). The DC power supply's positive (red) is connected directly to the positive of the water pump, while its negative (black) is connected to one of the pins on the secondary side of the relay switch. A blue Relay switch can be seen connected to the water pump as well (connected to the signal pin of the water pump from the Raspberry Pi which then goes to the pump). The signal from the Raspberry Pi is connected to the primary side of the relay which then goes to the negative of the pump that is connected on the secondary side of the relay. The water level sensor in the water trough (green) is connected to the signal pin from the Raspberry Pi as well. The ultrasonic sensor and the LED are also connected to signal pin from the Raspberry Pi as well as the servo motor. The ultrasonic sensor and the servo motor are mounted on the black box in the picture close to the water trough. The water level sensor, the ultrasonic sensor, the servo motor, the LED and the relay switch are all connected to the 5V from the Raspberry Pi and to the ground of the Raspberry Pi. The water pump alone is connected to the DC power supply and this is because the pump was drawing too much power from the Raspberry Pi and it was causing it to be switching off.

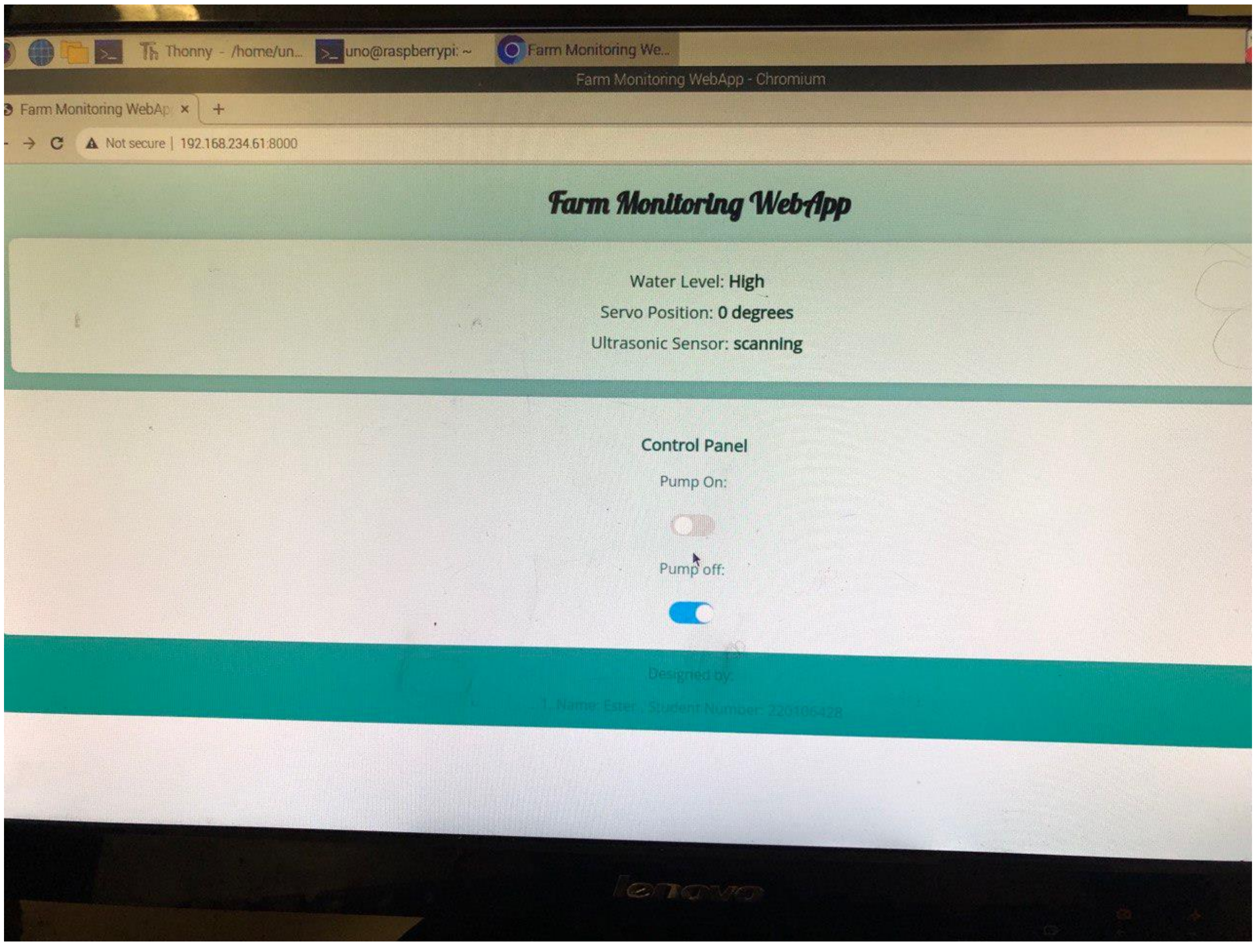
Figure 4: The circuit connected



The circuit in Figure 3 shows how the components are connected to their signal pins from the Raspberry Pi and how they are connected to the 5V and the ground of the Raspberry Pi.

Results & Discussions

Figure 5: The Mobile App 1



Farm Monitoring WebApp

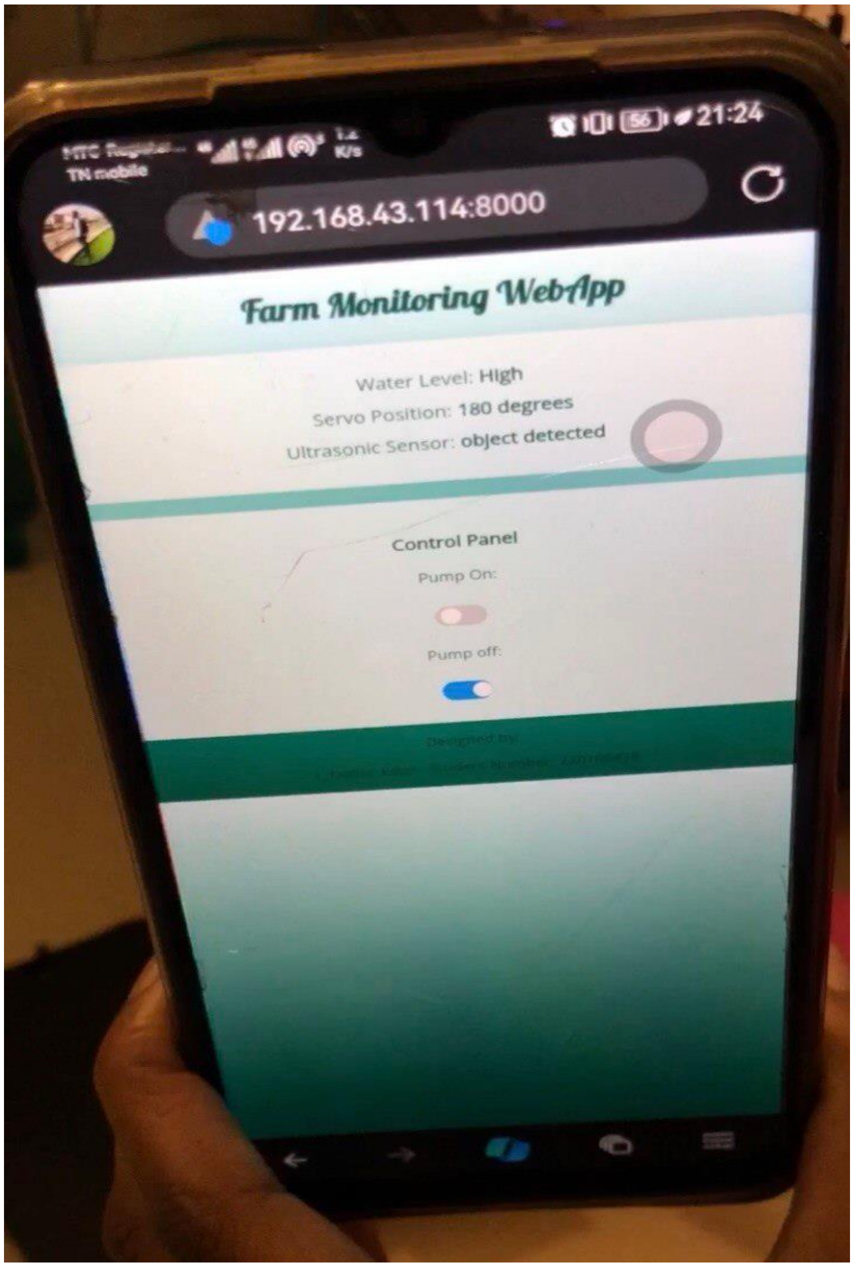
Water Level: **High**
Servo Position: **0 degrees**
Ultrasonic Sensor: **scanning**

Control Panel
Pump On:
Pump off:

Designed by:
1. Name: Ester , Student Number: 220106428

The picture in Figure 4 shows the Mobile App and all that it displays. The water level sensor, ultrasonic sensor and servo motor readings are all displayed on the Mobile App. When the water level sensor reads ‘HIGH’ as shown is Figure 4, then the ‘Pump off’ button automatically goes on so that the pump stops pumping water into the trough. When the ultrasonic sensor can’t detect any presence of animals it reads ‘scanning’ and the servo motor reads ‘0 degrees’ as shown in Figure 4. All the components with readings displayed on the Mobile App have delays.

Figure 6: The Mobile App 2



192.168.43.114:8000

Farm Monitoring WebApp

Water Level: High

Servo Position: 180 degrees

Ultrasonic Sensor: object detected

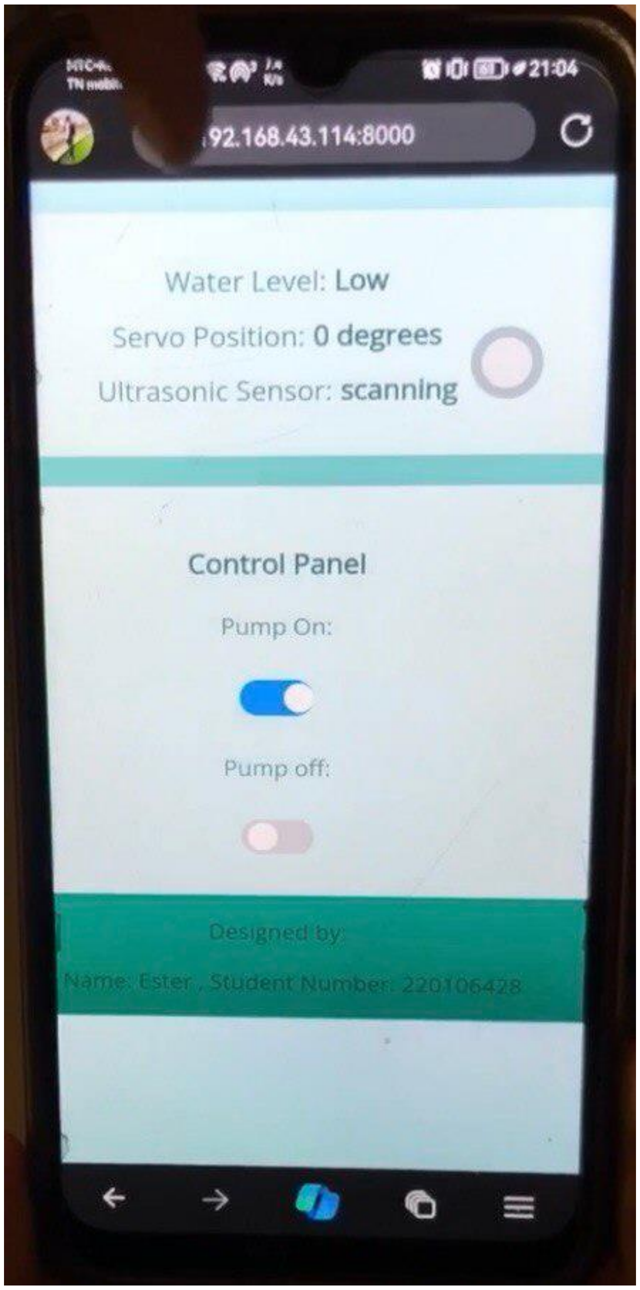
Control Panel

Pump On:

Pump off:

The picture in Figure 5 shows the readings of the ultrasonic sensor and servo motor when animal presence is detected.. When the ultrasonic sensor detects the presence of animals it displays ‘Object detected’ while the servo motor arm moves from its original position and reads ‘180 degrees’ as shown in Figure 5.

Figure 7: The Mobile App 3



Water Level: Low

Servo Position: 0 degrees

Ultrasonic Sensor: scanning

Designed by:

Name: Ester , Student Number: 220106428

The picture in Figure 6 shows the reading of the water level sensor when it reads ‘LOW’. When the water level sensor reads ‘LOW’ the farmer can turn on the ‘Pump on’ button so that the pump starts pumping water into the water trough that’s why the ‘Pump on’ button is on in Figure 6. During the pumping of water into the water trough, the farmer can stop the pump from pumping water by pressing the ‘Pump off’ button and this can be used in cases where the water level sensor fails to give readings to avoid wastage of water.

Conclusion

This project successfully designed, developed, and tested a Water Monitoring System for Farms with all the Key objectives met. This innovative system leverages the power of Internet of Things (IoT) technology to provide real-time monitoring of water levels in animal troughs and automates the refilling process. The system offers several key functionalities:

- Real-time water level monitoring: Ensures consistent access to fresh water for animals and prevents dehydration.
- Automated refilling: Eliminates the need for manual intervention and optimizes water resource utilization.
- Improved farm management: Provides farm managers with real-time data and remote control capabilities for informed decision-making.

Challenges faced

- **Internet Dependency for Remote Monitoring and Control**: The system remote monitoring and control is dependent on the Internet. While the system might function for basic monitoring locally, full functionality like remote access to data or control features would rely on a stable internet connection at the farm location. This could be a limitation in areas with limited or unreliable internet access.
- **IP Address-Based Control Limitations:** The design is IP address-based and an IP address-based control system offers localized security within the farm's network. However, this might restrict remote access and control functionalities from outside the network. For instance, the famer wouldn't be able to monitor or control the system from a smartphone app if you were away from the farm.
- **SG-90 Servo Motor Jitter**: The servo motor used for the design is an SG motor and it is prone to jitter. The SG-90 servo motor, while readily available, is known for exhibiting slight jittering movements. This could lead to inconsistencies or inaccuracies in the dispensing mechanism, particularly for salt licks.

Challenges overcome

During the design phase, a significant challenge involved merging individual component codes into a single, cohesive program. This required careful code integration and testing to ensure proper communication and functionality between all system elements.

Another hurdle encountered was the inability of the Raspberry Pi to handle the power demands of the pump directly. This issue was effectively addressed by introducing a separate 5V DC power supply and a relay. The relay acts as a switch, allowing the low-power Raspberry Pi signal to control the high-power pump without overloading the system.

Project Significance

The Water Monitoring System for Farms offers a valuable solution for modern farm management. It promotes animal welfare by ensuring consistent hydration and potentially providing essential salt licks. Additionally, the system optimizes resource utilization through automated refilling and empowers farm managers with remote monitoring and control capabilities, ultimately contributing to improved farm productivity and sustainability.

Future Enhancements

This project lays a strong foundation for further development. Future iterations could explore:

- Exploring the integration of a cellular communication module (e.g., cellular modem) into the system. This would allow for remote access and control functionalities regardless of the farm's internet availability, as long as cellular reception is present.
- For improved dispensing accuracy, considering using a higher-precision servo motor with reduced jitter. While these might be slightly more expensive, the increased accuracy could be crucial for salt lick dispensing applications.
- Integration of additional sensors to monitor factors like water quality, temperature, or animal activity. This comprehensive data can provide valuable insights into farm management.
- Implementing data analytics to identify trends and patterns in water usage and animal behavior, helping farmers to make informed decisions.
- Developing machine learning algorithms to optimize automated refilling schedules and resource allocation.
- Integrating the system with a reputable IoT platform like Amazon Web Services (AWS) IoT core, Microsoft Azure IoT Hub or Google Cloud IoT Core. This will enable remote access and control of the water monitoring system from anywhere with an internet connection, offering greater flexibility and convenience to farm managers.

By building upon this project's success and incorporating these potential advancements, the Water Monitoring System for Farms has the potential to revolutionize farm management practices, fostering a more efficient, sustainable, and animal-centric agricultural industry.

Works Cited

Logan Vandermark, H. M. (2023, September 25). *SOUTH DAKOTA STATE UNIVERSITY EXTENSION.* Retrieved from Water Monitoring Systems for Livestock: https://extension.sdstate.edu/water-monitoring-systems-livestock

Nalina Suresh, V. H. (2019). *ResearchGate.* Retrieved from Smart Water Level Monitoring System for Farmers: https://www.researchgate.net/publication/334662554_Smart_Water_Level_Monitoring_System_for_Farmers

Yash Panchal, B. S. (2023, July 27). *SSRN.* Retrieved from Smart Dustbin: Arduino Controlled Servo Motor and Ultrasonic sensor: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4513779

Appendices

```
import RPi.GPIO as GPIO

import time

from gpiozero import Servo, LED

from time import sleep

from datetime import datetime

from http.server import BaseHTTPRequestHandler, HTTPServer

import threading


GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)


# Define the GPIO pin numbers

servo_pin = 20

Pump = 21

water_pin = 18

TRIG = 23

ECHO = 24

LED_PIN = 25


# Create a servo object

servo = Servo(servo_pin)

pump = LED(Pump, active_high=False)  # Active low relay
```

```
ultrasonic_led = LED(LED_PIN)


# Setup ultrasonic sensor pins

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN (http://gpio.in/)) (http://gpio.in/)

GPIO.setup(water_pin, GPIO.IN (http://gpio.in/)) (http://gpio.in/)


# Global variables

water_level_high = False

servo_position = '0 degrees'

ultrasonic_state = 'scanning'


# Manual control flags

manual_control_pump = False

manual_control_led = False


# Web server configuration

host_name = '10.102.74.34'  # Replace with your Raspberry Pi's IP address

host_port = 8000


def check_water_level():

    global water_level_high

    water_level_high = GPIO.input(water_pin)

    return water_level_high


def ultrasonic_scan():
```

```
global servo_position, ultrasonic_state


while True:
    GPIO.output(TRIG, False)
    time.sleep(0.2)


    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)


    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()


    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)


    if distance <= 50:
        servo_position = '180 degrees'
        servo.value = 1  # Set the servo position to 1 (180 degrees)
        sleep(4)
        ultrasonic_state = 'object detected'
    else:
        servo_position = '0 degrees'
```

```
            servo.value = -1  # Set the servo position to -1 (0 degrees)
            ultrasonic_state = 'scanning'
        sleep(1)
def pump_control():
    global manual_control_pump

    while True:
        water_detected = check_water_level()
        if not water_detected and not manual_control_pump:
            time.sleep(13)
            if not water_detected:
                pump.on()
                time.sleep(2)
        elif water_detected and not manual_control_pump:
            time.sleep(0.1)
            if water_detected:
                pump.off()
        sleep(1)

class RequestHandler(BaseHTTPRequestHandler):
    def do_HEAD(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def _redirect(self, path):
        self.send_response(303)

        self.send_header('Content-type', 'text/html')

        self.send_header('Location', path)

        self.end_headers()


    def do_GET(self):

        global water_level_high, servo_position, ultrasonic_state


        water_level_status = 'High' if water_level_high else 'Low'

        pump_status = 'On' if pump.is_lit else 'Off'

        led_status = 'On' if ultrasonic_led.is_lit else 'Off'

html = f'''

        <!DOCTYPE html>

        <html>

        <head>

            <title>Farm Monitoring WebApp</title>

            <style>

                body {{

                    font-family: 'Open Sans', sans-serif;

                    text-align: center;

                    background: linear-gradient(to bottom, #e0f7fa, #00695c);

                    color: #004d40;

                }}

                h1 {{

                    font-family: 'Lobster', cursive;

                    color: #004d40;
```

```
}}
.status {{
    margin: 20px;
    padding: 20px;
    border-radius: 10px;
    background-color: #e0f2f1;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}}
.status p {{
    margin: 10px 0;
    font-size: 18px;
}}
.status .highlight {{
    font-weight: bold;
}}
.controls {{
    background-color: #e0f2f1;
    padding: 20px;
    border-radius: 10px;
    margin-top: 20px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}}
.controls .switch {{
    position: relative;
    display: inline-block;
    width: 50px;
    height: 25px;
    margin: 10px;
}}
.controls .slider {{
    position: absolute;
    cursor: pointer;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: #ccc;
    transition: .4s;
    border-radius: 25px;
}}
.controls .slider:before {{
    position: absolute;
    content: "";
    height: 20px;
    width: 20px;
    border-radius: 50%;
    left: 5px;
    bottom: 2.5px;
    background-color: white;
    transition: .4s;
}}
input[type="checkbox"]:checked + .slider {{
```

```
                background-color: #2196F3;

            }}

            input[type="checkbox"]:checked + .slider:before {{

                transform: translateX(25px);

            }}

            .indicator {{

                margin: 20px;

                padding: 20px;

                border-radius: 10px;

                background-color: #e0f2f1;

                box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

            }}

            .indicator span {{

                font-weight: bold;

            }}

        </style>

                                                                                <link                href="
(https://fonts.googleapis.com/css2?family=Open+Sans&family=Lobster&display=swap)https://fonts.goo
gleapis.com/css2?family=Open+Sans&family=Lobster&display=swap"
(https://fonts.googleapis.com/css2?family=Open+Sans&family=Lobster&display=swap)
rel="stylesheet">

        <meta http-equiv="refresh" content="6">

    </head>

    <body>

        <h1>Farm Monitoring WebApp</h1>

        <div class="status">

            <p>Water Level: <span class="highlight">{water_level_status}</span></p>

            <p>Servo Position: <span class="highlight">{servo_position}</span></p>
        <p>Ultrasonic Sensor: <span class="highlight">{ultrasonic_state}</span></p>

      </div>

      <div class="controls">

        <h3>Control Panel</h3>

        <form action="/" method="POST">

          <p>Pump on:</p>

          <label class="switch">

              <input type="checkbox" name="pump" value="pump" {'checked' if pump.is_lit else ''}
onchange="this.form.submit()">

            <span class="slider"></span>

          </label>

          <p>Pump off:</p>

          <label class="switch">

             <input type="checkbox" name="led" value="led" {'checked' if ultrasonic_led.is_lit else ''}
onchange="this.form.submit()">

            <span class="slider"></span>

          </label>

        </form>

      </div>

      <p>Designed by:</p>

      <p>1. Name: Ester , Student Number: 220106428</p>

    </body>

    </html>'''


    self.do_HEAD()

    self.wfile.write(html.encode('utf-8'))
    def do_POST(self):
        global manual_control_pump, manual_control_led

        content_length = int(self.headers['Content-Length'])
        post_data = self.rfile.read(content_length).decode('utf-8')

        if 'pump=pump' in post_data:
            pump.toggle()
            manual_control_pump = not pump.is_lit
        if 'led=led' in post_data:
            ultrasonic_led.toggle()
            manual_control_led = not ultrasonic_led.is_lit

        self._redirect('/')

def run(server_class=HTTPServer, handler_class=RequestHandler, port=host_port):
    server_address = (host_name, port)
    httpd = server_class(server_address, handler_class)
    print(f'Starting server on http://{host_name}:{port}')
    httpd.serve_forever()

if name == '__main__':
    try:
        # Start the sensor loops in separate threads
        threading.Thread(target=ultrasonic_scan, daemon=True).start()
        threading.Thread(target=pump_control, daemon=True).start()
        run()
    except KeyboardInterrupt:
        print('Server stopped')
        GPIO.cleanup()
```