Are Drones the Future of Agriculture?

By Prabhuansh Sadana and Jogeshwor Mundi

TABLE OF CONTENTS

PROBLEM

In modern society, many countries around the world have limited access to advanced farming equipment. This is mainly due to the cost of these equipment. In fact, agricultural drones can cost around 1,000 to 10,000 dollars if someone is looking for a drone with better sensors, cameras, and an increased payload. So, in order to find a solution to this problem, we wanted to see how we could make a budget friendly drone with an increased payload, quality cameras, and sensors so that people around the world who have limited access to this technology can have a better opportunity towards farming. However, once these technologies are accessed throughout the world, it may not be as sustainable and productive as compared to traditional farming practices. Farmers need to prepare their crops in a short period of time to ensure that the farmer can make more profits. In order to find a solution to this problem, however, we need to experiment and observe how long the drone takes to do a specific task (measure the soil moisture of plants) in order to conclude its efficiency. So, these 2 solutions not only allow farmers to have access to advanced farming technology but also make informed decisions about the efficiency of these technologies. With this information, we can conclude if drones are the future of agriculture.

With this in mind, we decided to form two questions that we would find a solution to with our prototypes:

1. Is it possible to create a budget friendly, cost effective drone with similar functions as to those in current society?
2. Is a drone able to perform better in terms of speed compared to traditional farming practices?

METHOD

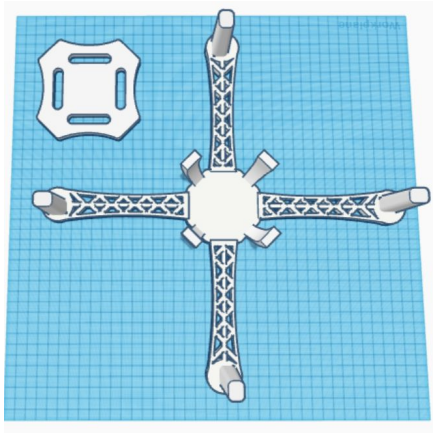
Prototype 1

Design and Programming

Design

Our first prototype was our very first attempt at creating the agricultural drone. The prototype consisted of 2 motor chips, known as the L289N motor drivers, and 4 DC Motors. We also used an Arduino Uno and a Raspberry Pi Zero W to control 2 of the motors individually, and an additional Arduino Uno to control the soil moisture sensor. This prototype unfortunately had no video capabilities and was lacking a very crucial component that many agricultural drones have that are currently in the agricultural industry. So, most of the footage shown from this prototype were from the ground and were not integrated into the model itself.

The chassis was created on a simple modeling software, Tinkercad, as we lacked the experience for any industrial modeling softwares. The model, though designed on Tinkercad, proved to be lightweight, durable, and efficient for our project. The quadcopter design did have many qualities and was used throughout the 3 prototypes we had created.



Programming

The drone was using 2 main coding languages, C++ (Arduino variation) and Python which was used in the Raspberry Pi. The Raspberry Pi in this project controlled two of the motors, while the Arduino Uno controlled the other two. The code was organized as follows:

Raspberry Pi Code (Python)

1. Define all wires and where they are at the microcontroller.
2. Create four sections for different code.

```
Motor1A = 23
Motor1B = 24
Motor1Enable = 25

Motor2A = 17
Motor2B = 27
Motor2Enable = 22

Motor3A = 5
Motor3B = 6
Motor3Enable = 13

Motor4A = 16
Motor4B = 20
Motor4Enable = 21
```

- **1. GPIOSTART**
  - This function is very important. This initializes the microcontroller and makes it ready to listen to commands.

- **2. DRONECONTROL**
  - This function is also very important. This function controls the drone and tells it when the motors that are supposed to be on and also when to turn off. (next slide)

```
import RPi.GPIO as GPIO
import time


GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
for pins in motor_pins:
    for pin in pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)

def turn_on_motors():
    for pins in motor_pins:
        GPIO.output(pins[0], GPIO.HIGH)
        GPIO.output(pins[1], GPIO.LOW)
        GPIO.output(pins[2], GPIO.HIGH)

def turn_off_motors():
    for pins in motor_pins:
        GPIO.output(pins[2], GPIO.LOW)

try:
    turn_on_motors()
    time.sleep(5)
    turn_off_motors()

except KeyboardInterrupt:
    turn_off_motors()

finally:
    GPIO.cleanup()
```

- **3. SUBMITDATA**
  - This function sends sensor data to the ground where the operators will be and will receive data about the weather, gyroscope, and air pressure that is around the microcontroller.
  - The microcontroller is at the bottom of the motors to make sure there is no interference from the motors blowing air into the sensor, therefore changing the result. Furthermore, the drone will try to stay in stable positions with the help of the built in gyroscope.

```
const int soilMoisturePin = A0;
const int dryThreshold = 600;
const int wetThreshold = 300;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int soilMoistureValue = analogRead(soilMoisturePin);
  Serial.println(soilMoistureValue);

  if (soilMoistureValue >= dryThreshold) {
    Serial.println("Soil is dry - needs water!");
  } else if (soilMoistureValue <= wetThreshold) {
    Serial.println("Soil is wet enough");
  } else {
    Serial.println("Soil moisture is within acceptable range");
  }

  delay(1000);
}
```

ARDUINO CODE (from the Arduino IDE)

1

```
// Motor A connections
int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B connections
int enB = 5;
int in3 = 7;
int in4 = 6;
```

2

```
  // Motors initially stopped
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

3

```
void loop() {
  // Run both motors forward for 5 seconds
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255); // Motor A speed (0-255),

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, 255); // Motor B speed (0-255),
```

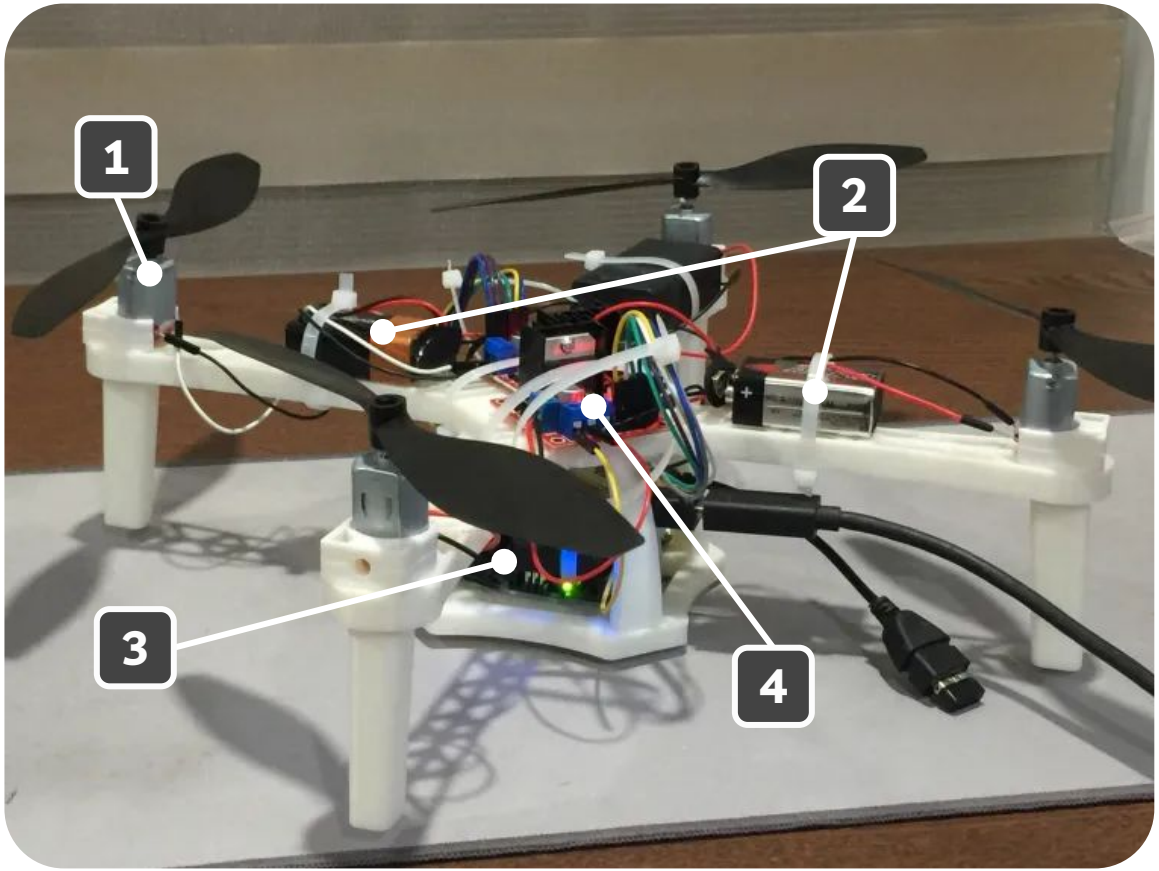
4

```
// Stop both motors after 5 seconds
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
analogWrite(enA, 0);

digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
analogWrite(enB, 0);
```

Building and Functionality

The construction of the agricultural drone consisted of the flight controllers at the bottom of the drone, the motor drivers at the top of the drone, and the 4 DC motors on each arm of the drone. A soil moisture sensor was also integrated near the bottom of the drone to allow the drone to obtain the soil moisture of the soil samples. Most of the pieces were connected using super glue, and the batteries and such were connected using zip-ties. To connect each of the components to each other, a system of wires was integrated. The wires connecting to the motors was not very strong, which was a strong source of error for this prototype.



1) Motors and Propellers
2) Batteries
3) Arduino and Raspberry Pi
4) Motor Drivers

Upon further testing of the drone, we did not previously anticipate the amount of load the DC motors would be able to carry. As a result, the drone was not able to achieve very much lift, and was not able to fly very well. The DC Motors that we bought for this project were unable to produce heavy amounts of lift for the drone to stay up from the ground for extended periods of time. Due to this issue, the drone could only reach low altitudes. The drone was able to hover for a few seconds and then would lose altitude relatively quickly. This iteration of the project proved to be budget friendly and had already passed the first part of our problem statement. However, due to the limited functionality of the drone, the drone was very slow and could not fly very well.

To conclude this section, Although this prototype did in fact function properly, there were major flaws in the wiring and maneuvering components of the drone. For one, the drone was not able to achieve much lift, as mentioned previously. This was mainly due to the inefficient motors that we had purchased for the prototype. Secondly, the drone was not able to hover in a single position, and was not very stable. In the future, we would like to add things such as gyroscopes to aid in the steady maneuvering of the drone, so that the drone could steadily obtain the soil moisture of plants. Finally, it was very difficult to keep the whole machine compact as the wires made it difficult for the propellers to spin without disturbing the connection.

Prototype 2

Design and Programming

Design

For this prototype, the design remained the same, however this iteration was a minor improvement as compared to our previous prototype. Our major flaw in our previous prototype was the lack of video footage. The new prototype was equipped with the Pi Camera so that the drone would be able to record footage from a far distance. Although the Pi Camera did work, it was useless as the drone could not reach high altitudes so that the camera could obtain footage.

Programming

PICAMERA

- This function allows the onboard camera to record for 5 minutes. We will check the footage recorded after the flight is concluded.

```python
import picamera
import time

camera = picamera.PiCamera()

camera.resolution = (640, 480)  # Adjust the resolution as needed

file_path = '/path/to/storage/dronevideofolder'  # Update the path as needed

try:

    camera.start_recording(file_path)


    camera.wait_recording(300)


    camera.stop_recording()

finally:

    camera.close()
```

Building, Testing, and Functionality

The functionality and results of the drone remained the same, the only difference obviously being the addition of the Pi Camera. The Pi Camera allowed us to receive video footage that we previously could not receive, and was overall very useful for our future prototypes. This prototype surpassed the first section of our problem once again, but still lacked the speed for the drone to be able to surpass human capabilities at a large scale.

Overall, although this iteration was successful in some areas, it did have many flaws that did affect the performance and accuracy of the drone itself. The code, design, etc. of this prototype remained the same, however, the only difference was the addition of the camera. We planned for our next prototype, aiming to make the flight performance and accuracy of the drone different in many aspects.

Prototype 3

Design and Programming

Design

For our last and current prototype, we decided to change many factors of the drone to allow the prototype to achieve lift effectively, remain stable throughout flight, and overall ease the wiring and other processes. For one, we decided to completely replace the DC motors with brushless motors, which allowed the propellers to reach staggering RPM's (revolutions per minute). Along with the brushless motors, we also needed to purchase ESC's which allow the motors to have a stable connection with our flight controllers. Furthermore, the Raspberry Pi, once used for the motors, was now changed to be used for the soil moisture sensor, and having the 2 Arduino Uno's control the motors. Finally, we decided to integrate a gyroscope (MPU 6050) into this prototype to allow the drone to be stable, and to connect 2 motors to the Arduino at one time. The soil moisture sensor and camera were in the same positions of the last 2 prototypes, however the programming for the soil moisture sensor had to be adjusted to accommodate for the changes we made in this prototype.

The chassis remained the same for this prototype as well. However, in order to connect the motors to the drone, adjustments would have to have been made. We had to use a wood piece to screw the motors on, and then connect that piece onto the drone, as the motors were quite bigger as compared to the drone's motor areas. The new design proved to be successful, as the new motors were powerful enough to generate the lift needed for the drone to hover in the air, and easily obtain the soil moisture.

Programming

Due to the change in the motors, new programming would have to been added to make the motors function. Furthermore, we also had to recode the soil moisture sensor with the Raspberry Pi now taking control of it. We also had to do many calculations to get the gyroscope correct and allow the drone to remain stable throughout flight. The code needed for our motors was now all Arduino C++.

Arduino Code (Motors)

- Define all libraries needed for our project (there is red lines because my Visual Studio Code was not updated).
- Define all the variables needed for the MPU-6050 to function (gyroscopes and accelerometers).
- The setup function that is required which initializes the drone for flight.

```
#include <Wire.h>
#include <Servo.h>
```

```
Servo right_prop;
Servo left_prop;

int16_t Acc_rawX, Acc_rawY, Acc_rawZ, Gyr_rawX, Gyr_rawY, Gyr_rawZ;
float Acceleration_angle[2];
float Gyro_angle[2];
float Total_angle[2];
```

```
void setup() {
  Wire.begin();
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(250000);
  right_prop.attach(3);
  left_prop.attach(5);

  time = millis();
  left_prop.writeMicroseconds(1000);
  right_prop.writeMicroseconds(1000);
  delay(7000);
}
```

This is the void Function tells commands for all ESCs to follow. Perform PID control (stabilizes drone during flight with MPU6050 chip).

```
d loop() {
timePrev = time;
time = millis();
elapsedTime = (time - timePrev) / 1000;

// Check current stage and perform actions accordingly
switch (stage) {
  case 0: // Full speed for 2 seconds
    if (millis() - stageStartTime < 2000) {
      throttle = 1050;
    } else {
      stageStartTime = millis();
      stage++;
    }
    break;
  case 1: // Half speed for 4 seconds
    if (millis() - stageStartTime < 4000) {
      throttle = 1050; // Half speed
    } else {
      stageStartTime = millis();
      stage++;
    }
    break;
  case 2: // Complete stop
    if (millis() - stageStartTime < 2000) {
      throttle = 1000; // Min speed
      // Set both motors to stop
      left_prop.writeMicroseconds(1000);
      right_prop.writeMicroseconds(1000);
    } else {
      stageStartTime = millis();
      stage = 0; // Reset to first stage
      delay(999999);
    }
    break;
}
```

```
oid performPIDControl() {
  Wire.beginTransmission(0x68);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(0x68, 6, true);

  Acc_rawX = Wire.read() << 8 | Wire.read();
  Acc_rawY = Wire.read() << 8 | Wire.read();
  Acc_rawZ = Wire.read() << 8 | Wire.read();

  Acceleration_angle[0] = atan((Acc_rawY / 16384.0) / sqrt(pow((Acc_rawX / 16384.0), 2) + pow((Acc_rawZ / 16384.0), 2))) * rad_to_deg;
  Acceleration_angle[1] = atan(-1 * (Acc_rawX / 16384.0) / sqrt(pow((Acc_rawY / 16384.0), 2) + pow((Acc_rawZ / 16384.0), 2))) * rad_to_deg;

  Wire.beginTransmission(0x68);
  Wire.write(0x43);
  Wire.endTransmission(false);
  Wire.requestFrom(0x68, 4, true);

  Gyr_rawX = Wire.read() << 8 | Wire.read();
  Gyr_rawY = Wire.read() << 8 | Wire.read();

  Gyro_angle[0] = Gyr_rawX / 131.0;
  Gyro_angle[1] = Gyr_rawY / 131.0;

  Total_angle[0] = 0.98 * (Total_angle[0] + Gyro_angle[0] * elapsedTime) + 0.02 * Acceleration_angle[0];
  Total_angle[1] = 0.98 * (Total_angle[1] + Gyro_angle[1] * elapsedTime) + 0.02 * Acceleration_angle[1];

  error = Total_angle[1] - desired_angle;

  pid_p = kp * error;

  if (-3 < error && error < 3) {
    pid_i = pid_i + (ki * error);
  }

  pid_d = kd * ((error - previous_error) / elapsedTime);

  PID = pid_p + pid_i + pid_d;
```

*Performs PID control which will be explained later.

Soil Moisture Code

- Import proper Python libraries that are needed.

```
import RPi.GPIO as GPIO
import time
```

- Find the correct threshold for the project and the soil moisture pin.

```
soilMoisturePin = 17

dryThreshold = 600
wetThreshold = 300
```

- Setup Raspberry Pi to receive soil moisture values from the sensor.

```
dryThreshold = 600
wetThreshold = 300

def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(soilMoisturePin, GPIO.IN)

def loop():
    try:
        while True:
            soilMoistureValue = analogRead(soilMoisturePin)
            print("Soil Moisture Value:", soilMoistureValue)

            if soilMoistureValue >= dryThreshold:
                print("Soil is dry.")
            elif soilMoistureValue <= wetThreshold:
                print("Soil is wet enough.")
            else:
                print("Soil moisture is within acceptable range.")

            time.sleep(1)

    except KeyboardInterrupt:
        print("Exiting...")

    finally:
        GPIO.cleanup()
```

- Read Analog Input sent from soil moisture sensor to Raspberry PI

```
f loop():
try:
    while True:
        soilMoistureValue = analogRead(soilMoisturePin)
        print("Soil Moisture Value:", soilMoistureValue)

        if soilMoistureValue >= dryThreshold:
            print("Soil is dry.")
        elif soilMoistureValue <= wetThreshold:
            print("Soil is wet enough.")
        else:
            print("Soil moisture is within acceptable range.")

        time.sleep(1)

except KeyboardInterrupt:
    print("Exiting...")

finally:
    GPIO.cleanup()
```

- Run both functions at the end of the code

```
if __name__ == '__main__':
    setup()
    loop()
```

Building, Testing, and Functionality

The organization of the new drone had to be majorly altered, as new components such as the LiPo batteries and the ESC's had to be stored. The LiPo batteries were on the bottom of 2 of the drone's arms, while the ESC's were located on the top of each arm. One Arduino was stored at the bottom of the drone, alongside the Arduino batteries, and one Arduino and the Raspberry Pi were stored on the top of the drone. Overall, this design proved to be successful in our testing and overall stability.

The functionality of the drone had completely changed. With the new additions of the brushless motors and the gyroscope, the flying process much smoother, and allowed the drone to make more precise movements that once couldn't be performed. Our new drone, although somewhat heavy, could achieve lift and could easily hover in the air, allowing us to accurately obtain the soil moisture of plants without any interruptions. However, the testing phase of the drone was somewhat different. The ESC's had to be calibrated every now and then or the motors would malfunction. The calibration was a huge part of the drone, as without calibrating the ESC's, the drone would not have been able to fly.

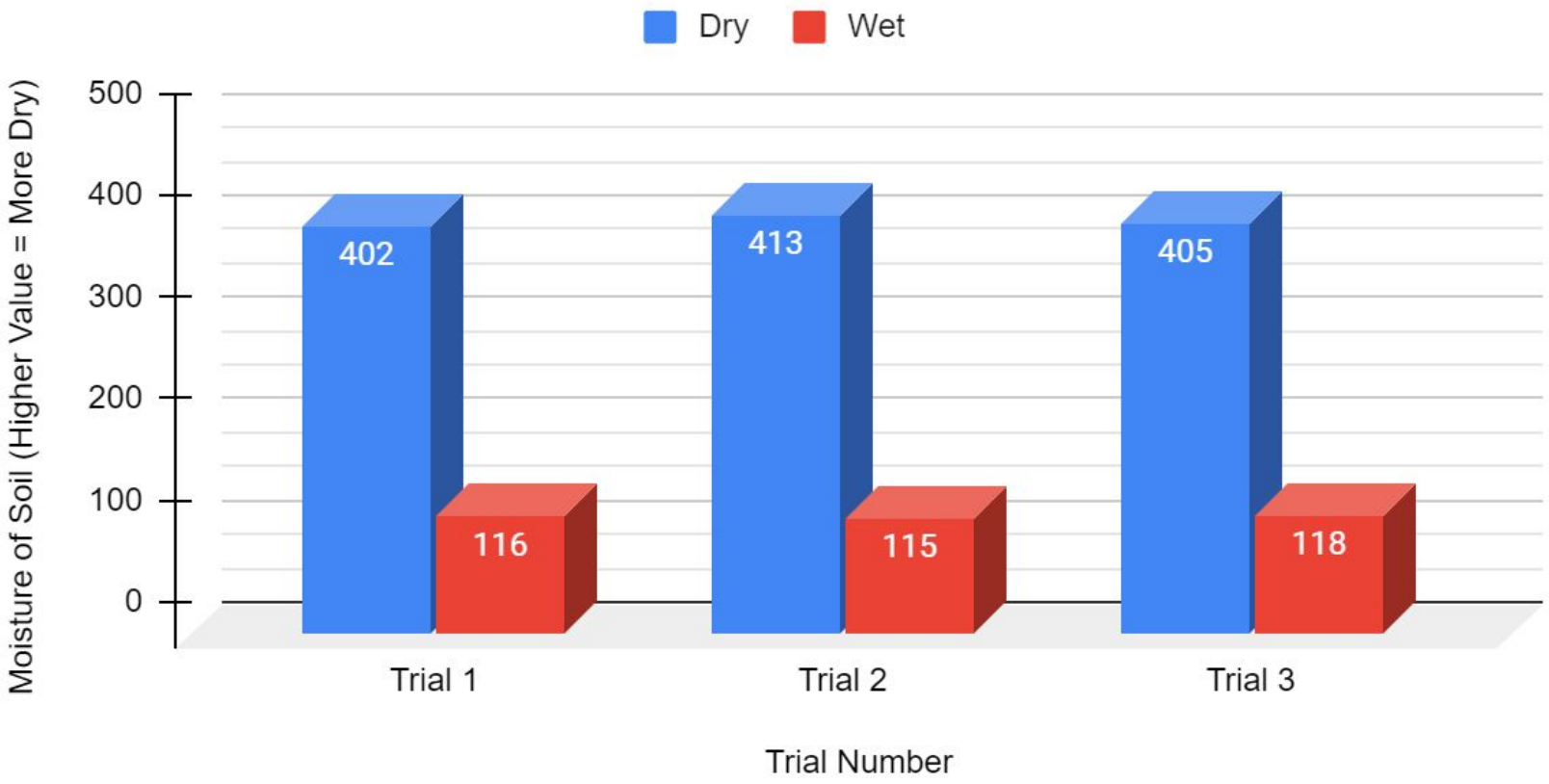
Overall, this prototype proved to be the most successful out of the 3, and we have surpassed both of our problem statements with the new prototype. The agricultural drone is not only easy to automate, but also easy to maneuver, as the gyroscope is able to control the drone and maintain its balance.

ANALYSIS

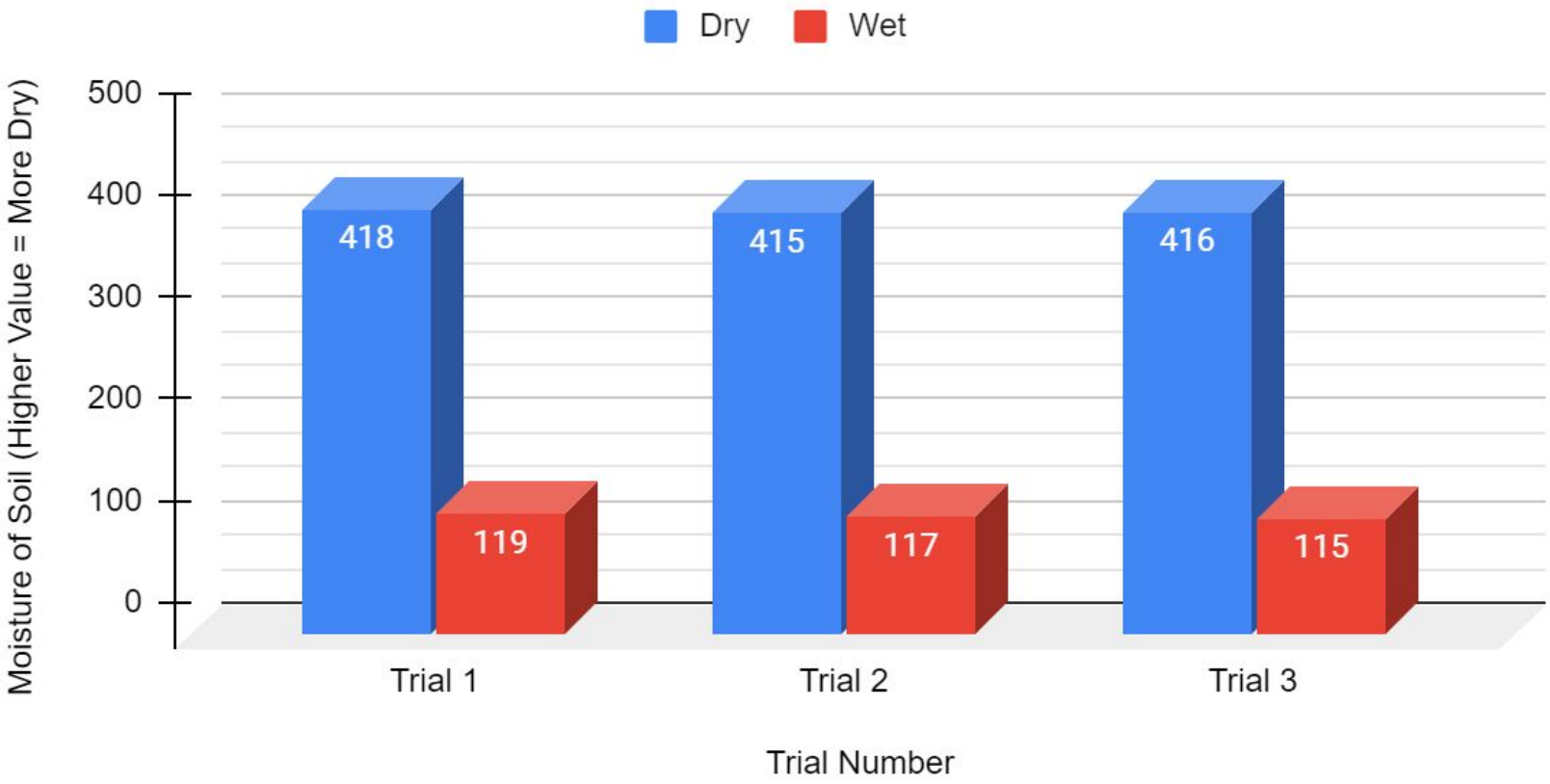
The testing phase of the drone will consist of the drone flying in the air, hovering, obtaining the soil moisture of the plant via the soil moisture sensor near the bottom of the drone, and then landing. We will also be taking a 'ground truth test', which is when the soil moisture sensor is tested without the drone to ensure that the drone provides accurate and informative data. We will be using this test to represent our traditional farming method, or the method used by humans. We will be timing both processes to see how long the drone takes to measure the soil moisture of the plants, compare it to our ground truth tests taken previously, and to conclude if the drone was more efficient than without the drone so that we can conclude if the drone is able to perform better in terms of speed compared to traditional methods. Finally, we will be testing the soil moisture for 3 plants and 3 trials for each, each trial testing both dry and wet soil.

Each of the prototypes gave us the same soil moisture values, as shown below:

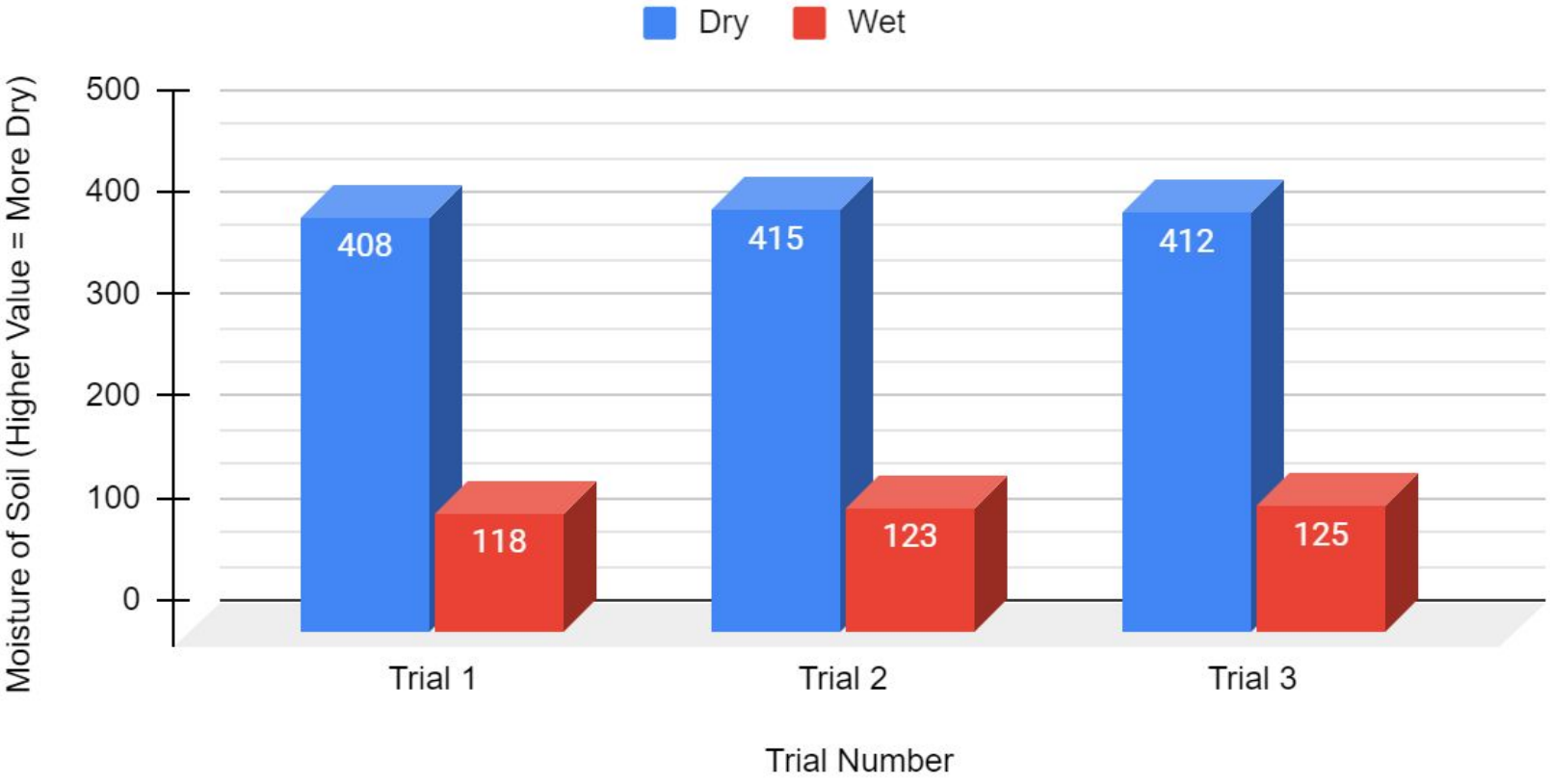
Ground Truth Test and Drone Test Soil Moisture Sensor Sample A Results



Ground Truth Test and Drone Test Soil Moisture Sensor Sample B Results



Ground Truth Test and Drone Test Soil Moisture Sensor Sample C Results



SAMPLE 1	Dry	Wet
Trial 1	402	116
Trial 2	413	115
Trial 3	405	118

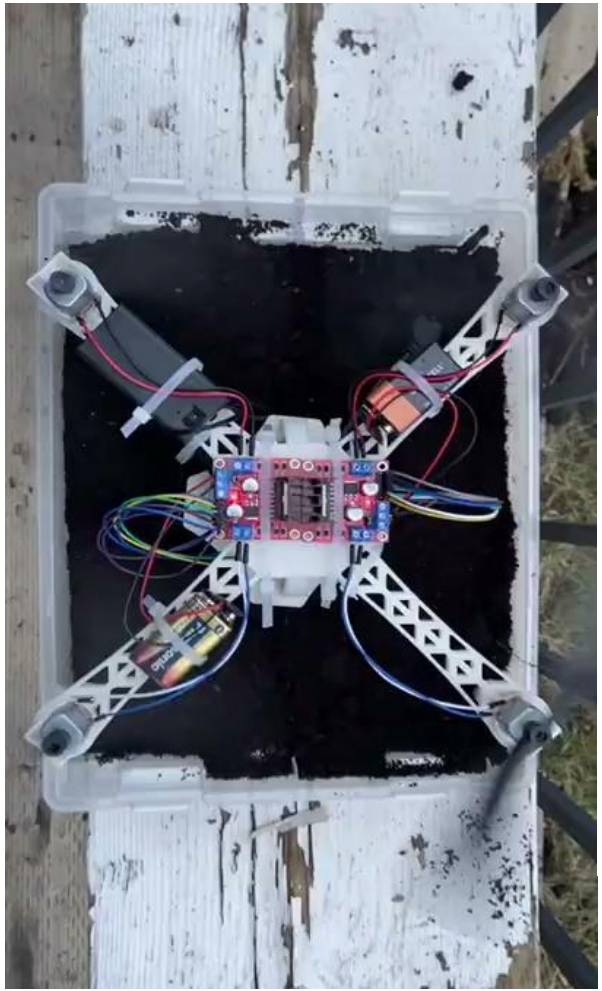
SAMPLE 2	Dry	Wet
Trial 1	418	119
Trial 2	415	117
Trial 3	416	115

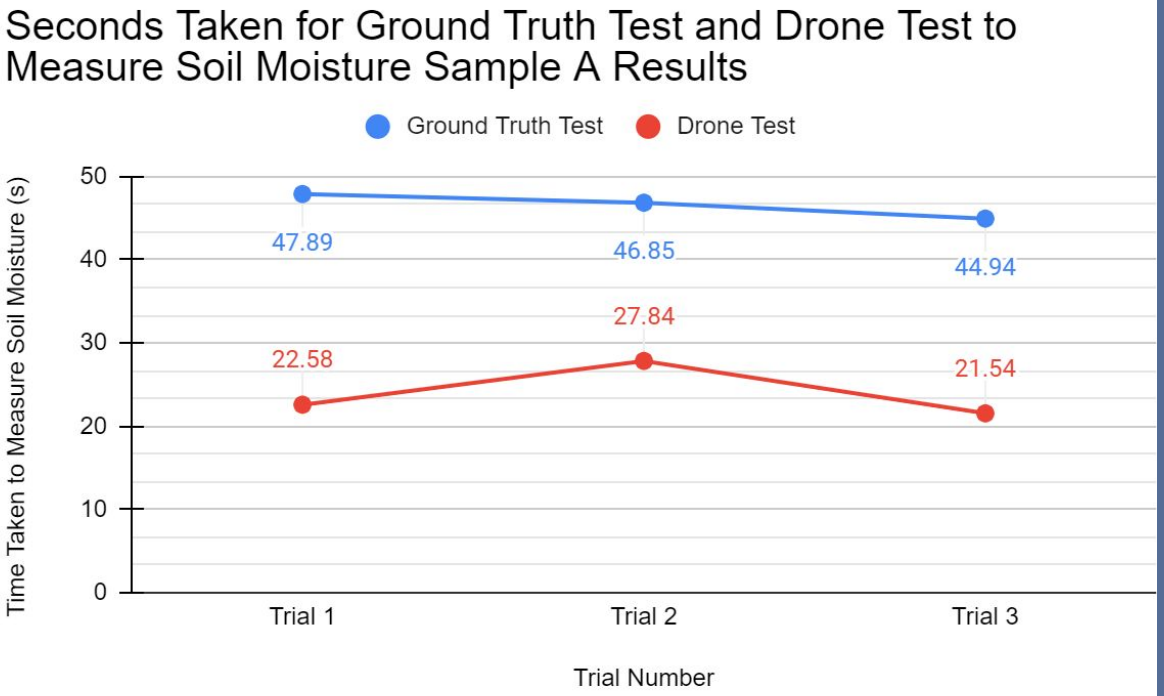
SAMPLE 3	Dry	Wet
Trial 1	408	118
Trial 2	415	123
Trial 3	412	125

Prototypes 1 and 2

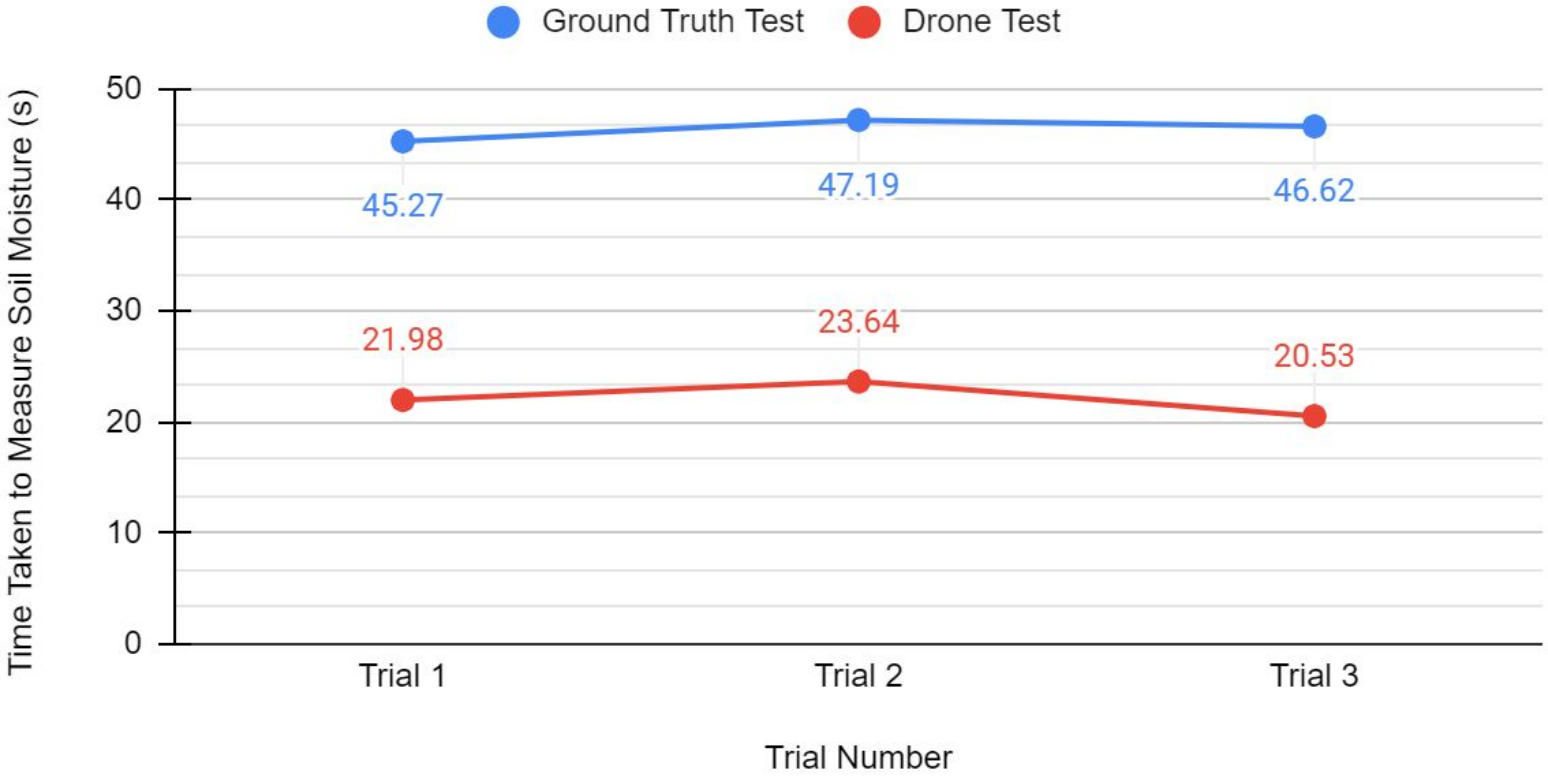
Prototypes 1 and 2 had the same results, as little to no changes were made regarding the motors and overall functionality (other than the camera). Our data from prototypes 1 and 2 showed that the drone was about 50% faster as compared to the ground truth tests. However, the drone did waste time as it took a while to get the drone off the ground and to maintain its hovering position. With this information, we decided to completely transform prototype 3 into something completely new, with many powerful motors that would help us to achieve that lift in a short amount of time.

Despite the drone lacking in generating lift accordingly, the results were still informative and allowed us to come to a solid conclusion - the drone was cost-efficient and was relatively cheaper than industrial agri-drones, however the drone, although faster than human methods, was still quite slow and could be much faster with more enhancements. The results are shown below.

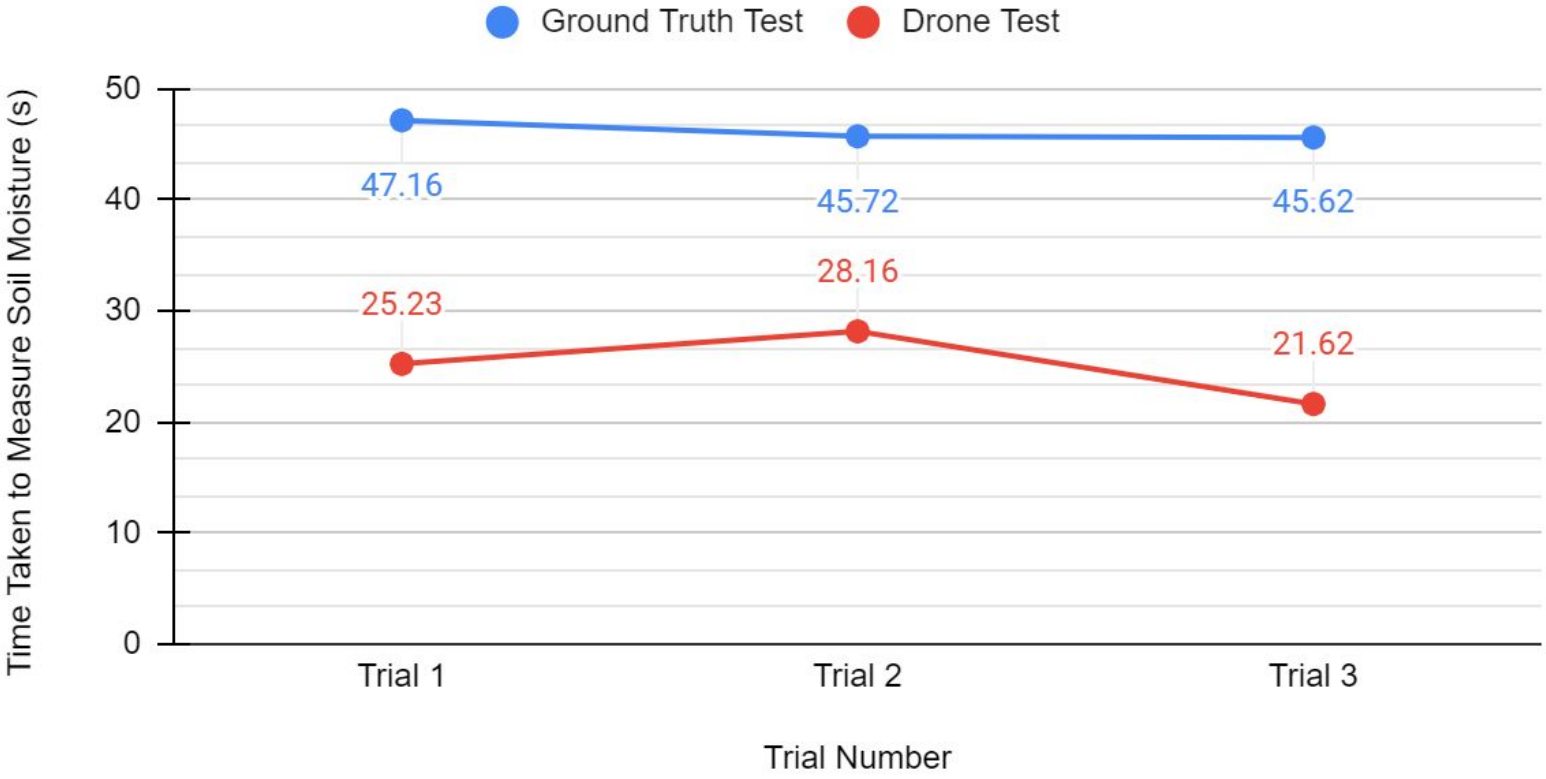



Seconds Taken for Ground Truth Test and Drone Test to Measure Soil Moisture Sample A Results
Ground Truth Test
Drone Test
Time Taken to Measure Soil Moisture (s)
50
40
30
20
10
0
47.89
46.85
44.94
27.84
22.58
21.54
Trial 1
Trial 2
Trial 3
Trial Number

Seconds Taken for Ground Truth Test and Drone Test to Measure Soil Moisture Sample B Results



Seconds Taken for Ground Truth Test and Drone Test to Measure Soil Moisture Sample C Results



SAMPLE 1	Ground Truth Test	Drone Test
Trial 1	47.89	22.58
Trial 2	46.85	27.84
Trial 3	44.94	21.54

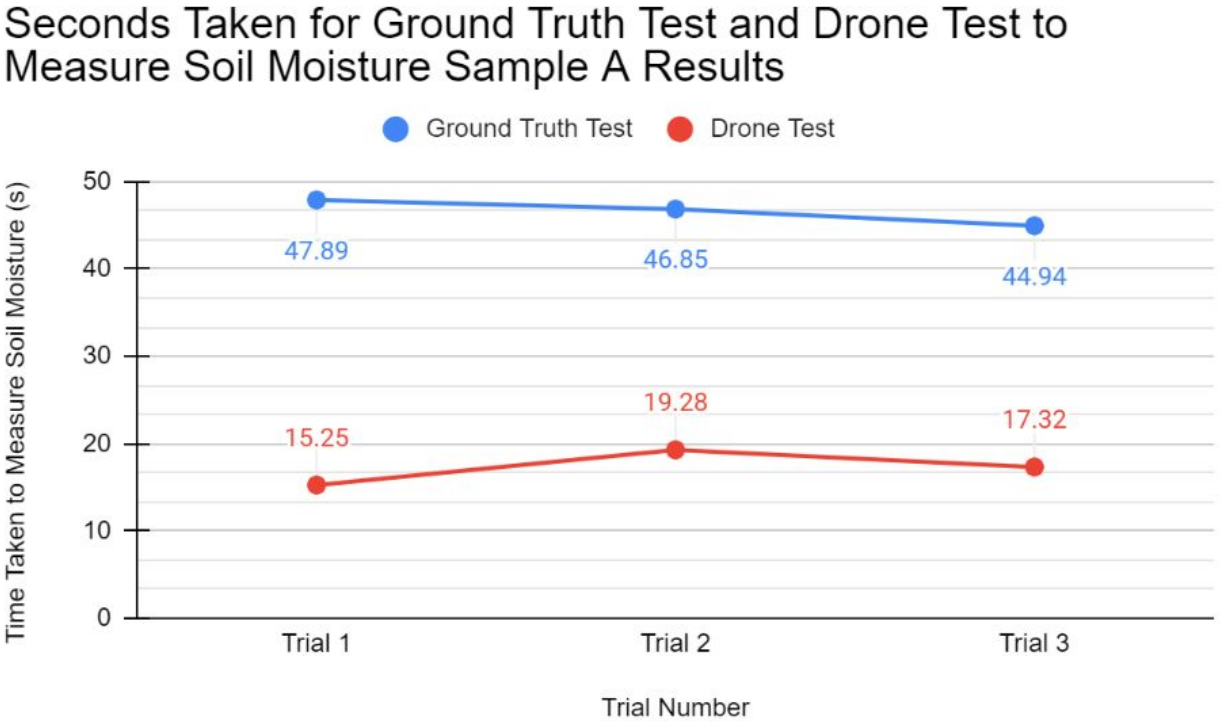
SAMPLE 2	Ground Truth Test	Drone Test
Trial 1	45.27	21.98
Trial 2	47.19	23.64
Trial 3	46.62	20.53

SAMPLE 3	Ground Truth Test	Drone Test
Trial 1	47.16	25.23
Trial 2	45.72	28.16
Trial 3	45.62	21.62

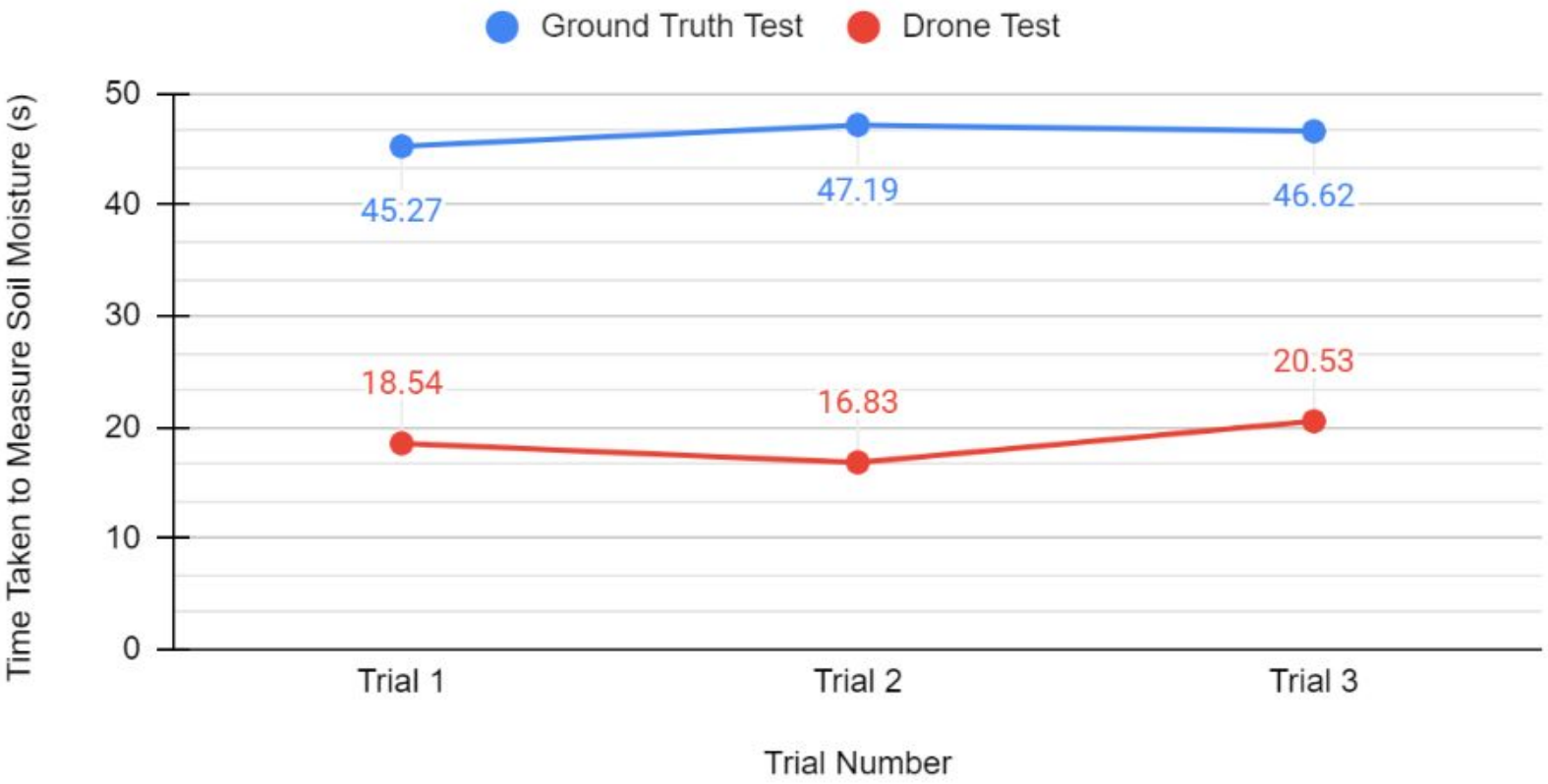
Prototype 3

Prototype 3 took much less time to measure the soil moisture of the soil as compared to the previous prototypes. The drone proved to be approximately 75% faster as compared to the ground truth tests. The drone took much less time to generate lift, and as a result, significantly decreased the time needed to hover in the air. Our new prototype showed that, through the use of drone technology, people can have a chance at accessing budget-friendly, advanced farming equipment so that they can be able to save time and money on expensive farming equipment. To conclude, our prototype successfully passed our initial problem statements, and overall was efficiently able to gather reliable and accurate data that we could use to innovate and expand the world of farming with.

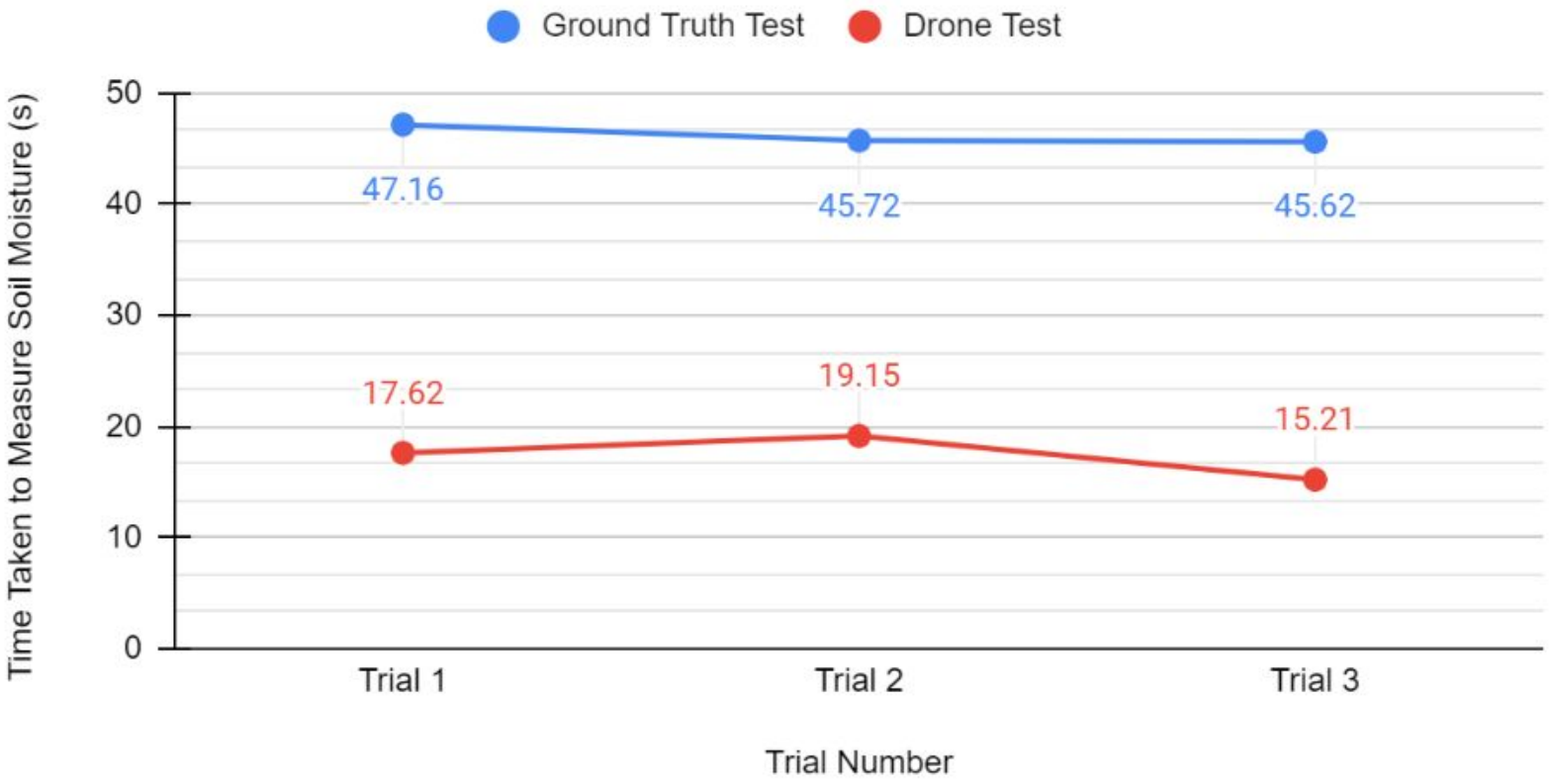
Below is the data collected for prototype 3 (keep in mind that the same ground truth test values were used for these ones as well).


Seconds Taken for Ground Truth Test and Drone Test to Measure Soil Moisture Sample A Results
Ground Truth Test
Drone Test
Time Taken to Measure Soil Moisture (s)
50
40
30
20
10
0
47.89
46.85
44.94
15.25
19.28
17.32
Trial 1
Trial 2
Trial 3
Trial Number

Seconds Taken for Ground Truth Test and Drone Test to Measure Soil Moisture Sample B Results



Seconds Taken for Ground Truth Test and Drone Test to Measure Soil Moisture Sample C Results



SAMPLE 1	Ground Truth Test	Drone Test
	47.89	15.25
Trial 2	46.85	19.28
Trial 3	44.94	17.32

SAMPLE 2	Ground Truth Test	Drone Test
Trial 1	45.27	18.54
Trial 2	47.19	16.83
Trial 3	46.62	20.53

SAMPLE 3	Ground Truth Test	Drone Test
Trial 1	47.16	17.62
Trial 2	45.72	19.15
Trial 3	45.62	15.21

CONCLUSION

1. Yes, it is possible to create a budget friendly drone with similar functions as those in current society, as demonstrated in our experiment. In current society, drones are used to scout plants and monitor health, monitor livestock, release pesticides, etc. Our project showed that it is possible to create budget friendly drones with increased payloads, quality cameras and sensors, and shows that the future of agriculture is rapidly approaching, and that farmers around the world will soon have access to this groundbreaking technology.
2. Yes, a drone is able to perform better in terms of speed as compared to traditional farming methods. Traditional farming methods mainly relied on the use of manual labor and organic pesticides, however, our drone shows that using more advanced farming methods will not only have an increased benefit on the crops, but also the farmers themselves. Our drone has the ability to work autonomously, meaning that farmers do not need to control the drone. Also, the drone is able to do all the functions the farmer requires 24/7, as the drone does not need breaks and such. So, with our drone, farming will improve in many aspects and will be incredibly efficient.

CITATIONS

Advantages of drones. (2021, May 30). COPTER.BG. Retrieved November 10, 2023, from https://www.copter.bg/en/blog/advantages-of-drones-and-drone-technology

Ahirwar, S. (2019, January 10). *Application of Drone in Agriculture*. https://www.researchgate.net/profile/Srinivas-4/publication/331017387_Application_of_Drone_in_Agriculture/links/5c6aacb04585156b57036824/Application-of-Drone-in-Agriculture.pdf

Allain, R. (2017, May 19). *The Physics of How Drones Fly*. WIRED. https://www.wired.com/2017/05/the-physics-of-drones/

Design and Development of Drone for Agricultural Applications. (n.d.). IJLEMR. http://ijlemr.com/papers/volume2-issue7/18-IJLEMR-22328.pdf

Drone Technology In Agriculture. (n.d.). Croptracker. Retrieved November 10, 2023, from https://www.croptracker.com/blog/drone-technology-in-agriculture.html

Dupuis, A. (2023, June 12). *Environmental Impact of Traditional Farming: 5 Effects*. Eden Green. https://www.edengreen.com/blog-collection/environmental-impact-of-traditional-and-vertical-farming-2021-report

Farm With A View: How Drone Technology Is Taking Agriculture To A New Level. (2023, February 23). Forbes. Retrieved November 10, 2023, from https://www.forbes.com/sites/stevensavage/2023/02/23/farm-with-a-view-how-drone-technology-is-taking-agriculture-to-a-new-level/?sh=2f05b3804962

Jao, E. (2023, July 31). *Traditional vs. Modern Farming Techniques: A Comparative Analysis*. Milk and Honey Ranch. https://milkandhoneyranch.com/gardening-and-farming/traditional-vs-modern-farming-techniques-a-comparative-analysis/

Johnston, M. (2023, September 7). *Thrust and Newton's Third Law in Aviation - Cal Aero Blog*. California Aeronautical University. https://calaero.edu/newtons-third-law-in-aviation/

Kamprad, D. (n.d.). *Urban Farming vs Traditional Farming: What's the Difference?* Impactful Ninja. https://impactful.ninja/urban-vs-traditional-farming-differences/

McSwine, D. (2023, November 7). *Implementation of drone technology for farm monitoring & pesticide spraying: A review*. https://www.sciencedirect.com/science/article/pii/S2214317322000087

McSwine, D. (2023, November 7). *Traditional Farming Practices and Its Consequences*. https://link.springer.com/chapter/10.1007/978-3-030-61010-4_6

STEM LEARNING: Advanced Air Mobility: The Science Behind Quadcopters Reader—Student Guide. (n.d.). NASA. https://www.nasa.gov/wp-content/uploads/2020/05/aam-science-behind-quadcopters-reader-student-guide_0.pdf