This project is a partnership with Geneva High School. We are honored to have worked with Ms.Walker and students at Geneva High School. Honorable mention and a sincere thank you to the Teaching Assistants of MAE 4220/4221: Yixin Zeng, Yichen Guo, and Even Leong.

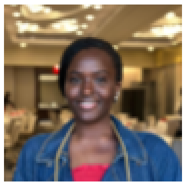
STEM

SCIENCE ● TECHNOLOGY ● ENGINEERING ● MATHEMATICS

Education Through IoT Plant Monitoring, May 2025

K. Max Zhang

Professor
Sibley School of Mechanical and Aerospace Engineering
Cornell University

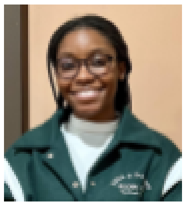


Cira Diop

Undergraduate Student
Sibley School of Mechanical and Aerospace Engineering
Cornell University

Brandon Feraud-Solorzano

Undergraduate and Graduate Student
Sibley School of Mechanical and Aerospace Engineering
Cornell University



Miriam Moonga

Undergraduate Student
Sibley School of Mechanical and Aerospace Engineering
Cornell University

Aya Mtume

Undergraduate Student
Sibley School of Mechanical and Aerospace Engineering
Cornell University

CornellEngineering
Sibley School of Mechanical and Aerospace Engineering

GENEVA HIGH SCHOOL

Table of Contents

Executive Summary

This project applies engineering design principles, sensor selection and deployment, wireless communication, and data quality management to develop an Internet of Things (IoT) Plant Monitoring Enclosure aimed at enhancing STEM education and generating a positive societal impact. The primary function of the system is to monitor and analyze plant growth conditions using a sensor-integrated IoT framework.

Many schools, particularly in underserved communities, lack access to experiential learning opportunities due to systemic underinvestment. Our project addresses this gap by modeling a low-cost, scalable solution that integrates hands-on technology with classroom curricula, engaging students in plant biology, environmental science, and data-driven inquiry.

The enclosure features three core components: a wooden lid that controls lighting, a transparent plant chamber for environmental observation, and a water compartment for soil moisture management. Sensors track soil moisture, humidity, temperature, and plant height, with data transmitted wirelessly via LoRaWAN and visualized through a remote dashboard. This design was developed in collaboration with Geneva High School. We worked closely with community partners Kirsten Abbott, a former AP Environmental Science teacher, and Shelley Walker, a botany teacher, to align the system with classroom goals and student interests. The final prototype was deployed in Ms. Walker's classroom, where students use it alongside their environmental botany curriculum.

Beyond this implementation, the project is intended to serve as a framework for broader use of sensor-based technologies in STEM education. The complementary instructional manual we've created, along with the IoT Plant Monitoring Enclosure, provides a replicable model that can be adapted to other subject areas to promote experiential, technology-enhanced learning across diverse educational settings.

Social Context

According to IoT Analytics[1], as of 2023, there are 16.6 billion IoT-connected devices, a figure that is projected to increase to 18 billion by the end of 2025. IoT is a rapidly growing industry and has been successfully integrated into our daily lives in many ways. In our phones, our homes, our cars, and our education systems, IoT has seamlessly made its way into our lives and those around us. Throughout this course, we have been able to gain a greater understanding, or even our first exposure to, the Internet of Things, or IoT. By going through extensive training labs and being able to successfully send temperature sensor data over the internet. As we moved through this course, our deeper understanding has allowed us to want to share this knowledge, especially with those who can build a future with this knowledge and potentially have little to no exposure to IoT. Science, technology, engineering, and math, or STEM for short, are integrated through all facets of human life. Job opportunities in STEM are increasingly prevalent, as we grow into a "smarter" world. According to studies conducted by the American University School of Education[2], exposure to STEM in early childhood education has become increasingly important and can promote ongoing academic success, preparing students to involve themselves in technology-focused professional atmospheres. STEM education is not simply about increasing knowledge in these specific fields, as it helps build better problem-solving skills, as well as encourages analysis and new ways of thinking. As these skills are essential to all facets of life, childhood/adolescent STEM education is incredibly important. According to a UC Irvine study, early math skills were the most consistent predictive measure of future academic success among K-5 students, showing the importance of STEM education overall and how it can be helpful to all students.

As students participate in an increasingly "smart" world, we find that it is very important for them to have access to some of the knowledge that we have obtained ourselves. Students may have smart homes, smartphones, smartwatches, Alexa, etc., but not fully understand how IoT has changed our world. During the COVID era, there was a global push towards "smart education," meaning using technology like Zoom, Google Meet, paperless work, etc., to make education happen for everyone from their homes. As explained by the Association for Computing Machinery[3], since "smart education" is not location or time-dependent, it minimized operational costs such as traveling for students and administrators, but increased initial costs to integrate and install smart systems. This created a greater course offering and teaching capabilities, as anyone can now operate from anywhere for anyone. This also created greater access to education, especially for students with disabilities or those who have difficulty traveling to places of education. With greater accessibility comes greater outreach and greater potential to expose students to different fields, including technological ones. However, there are still barriers that prevent students from having a comprehensive STEM education and exposure. Rural areas usually lack access to broadband, which is a high-speed, always-on internet connection that is significantly more efficient and faster than dial-up internet. Due to a lack of access, students in rural areas usually face more challenges when it comes to digital literacy and exposure to STEM education, specifically in the technological field. As explained before, access to STEM education is incredibly important in today's age. This is why we feel it is incredibly important to engage with the students in a way that they can better understand the world that surrounds them while also building upon knowledge that they already possess. Our idea is to create a fun, plant-growing activity that engages the students and allows them to apply their environmental science skills differently. The students will be observing the growth of a plant, sending data from temperature, moisture, and distance sensors in an enclosure that we create. This data will allow the students to detect when a plant needs to be watered or have the enclosure temperature altered based on the data received from the sensors. The students will be able to gain familiarity with basic circuits and hardware, as well as gain an understanding of the software that we will compile for our data collection.

Our community partner, Kirsten Abbott, is a former Geneva High School Chemistry and AP Environmental Science teacher, and has connected us to her colleague Shelley Walker, who teaches environmental botany at Geneva High School. Our project is focused on plant life, which is extremely relevant to their coursework and experiences thus far in the semester, hopefully making our project both worthwhile and impactful to the students. Access to STEM-related programs through Geneva High School is also relevant to this project, as the STEM-based clubs are elective. These clubs include the Green Club, Science Club, and Math Club. Our project is unique, as it gives students the opportunity to interact with these principles within the classroom as opposed to outside of it in elective clubs. It is incredibly important that students are allowed to have hands-on experience with STEM in their classrooms for those who are not sure if it's even an interest to them. As we have partnered with Mrs. Walker's environmental botany class, our project allows for greater opportunities for students to build a passion about STEM and see how it impacts their day-to-day lives. Hopefully, through this relevant and fun project, students will gain interest in STEM, specifically IoT, and be introduced to new learning principles that they can build upon in the future.

---

[1] Sinha, Satyajit. "State of IOT 2024: Number of Connected IOT Devices Growing 13% to 18.8 Billion Globally." *IoT Analytics*, 10 Feb. 2025, iot-analytics.com/number-connected-iot-devices/#:~:text=State%20of%20IoT%202023:%20Number,Sinha%20on%20May%2024%2C%202023.

[2.] "Why Is STEM Important in Early Childhood Education? Understanding Child Development and Learning: American University." *School of Education Online*, 15 Apr. 2024, soeonline.american.edu/blog/stem-in-early-childhood-education/#:~:text=Lifelong%20Benefits%20of%20Early%20STEM%20Education&text=For%20example%2C%20a%20study%20by,kindergarten%20to%20fifth%20grade%20students.

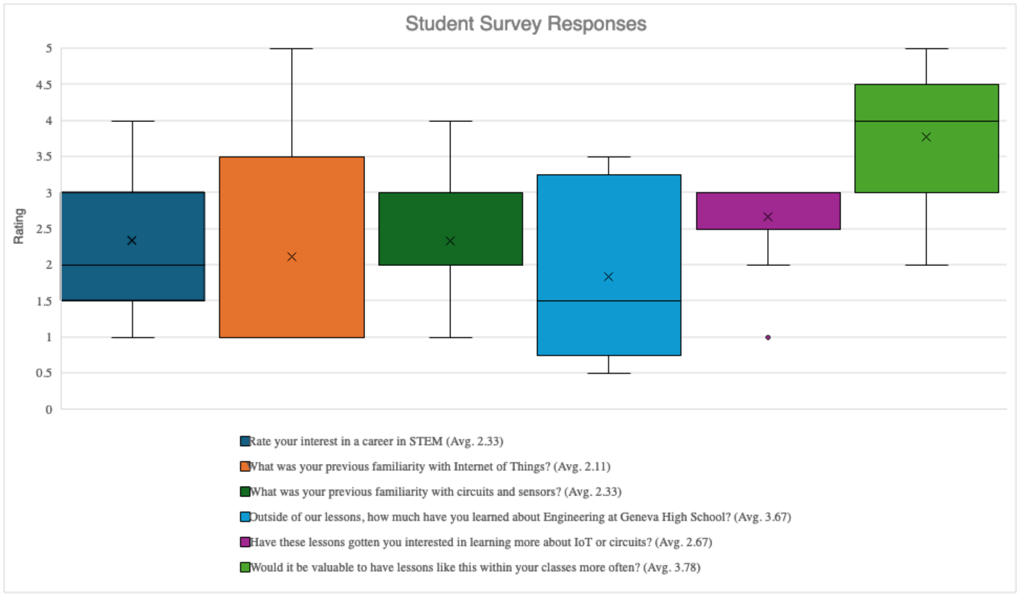
[3] Afzal Badshah, Anwar Ghani, Ali Daud, Ateeqa Jalal, Muhammad Bilal, and Jon Crowcroft. 2023. Towards Smart Education through Internet of Things: A Survey. ACM Comput. Surv. 56, 2, Article 26 (February 2024), 33 pages. https://doi.org/10.1145/3610401

Review of Related Works

In previous years, there has been considerable progress in the integration of IoT in high schools, but the state of IoT education for highschool students leaves much to be desired.

In 2023, the IoT for STEM team focused on educating students on the principles of how IoT works and having them build circuits. Their idea was to have a club in which students could treat IoT as an extracurricular activity, hoping this would encourage the formation of a more IoT-focused curriculum.

The 2024 STEM team wanted to focus more on increasing high school students' interest in STEM and having them work more closely with IoT components like sensors and circuits. They also wanted to emphasize the relevance of IoT to the students by having them integrate it into one of their courses. The team chose chemistry and designed a water quality monitoring device. Their design featured a turbidity sensor and a floatable casing to house their circuits. Because of the nature of their project (building a casing, putting the sensors and circuits together), they were able to involve the students more than the 2023 team. In addition to explaining *how* the sensors worked, they further explained their relevance for a real application the students were familiar with (turbidity, a chemistry concept), which was a step forward from the previous team. They also collected data and had this data relayed to the students at fixed time intervals, and the students were able to monitor the water quality in their school pond where the water quality monitor was placed. After they completed their project, they surveyed the students to gauge their interest in STEM. Their data showed that the students were more positive about learning more about STEM after they had participated in the project.



STEM Students 2024 Survey Chart showing average interest in STEM among 9 surveyed students

The past two teams have focused on integrating IoT specifically in Geneva High Schools, but the overall goal is to have IoT integrated in high schools as far as possible. Different groups have made several efforts to achieve this goal. For example, LocoRobo is a company that teaches STEM concepts to high school students using IoT[1]. They teach students concepts such as coding and robotics through hands-on learning experiences and have their lessons tailored towards different age groups. Rugged Telemetry is another example of an organization implementing a STEM IoT program. They have programs for high school and college students and use hands-on experience-based teaching (providing kits of mechanical components, for example) to get students involved and interested in IoT[2].

A larger-scale initiative with a mission aligned with ours is the Center for Research on Programmable Plant Systems (CROPPS), a National Science Foundation Science and Technology Center led by Cornell University. CROPPS aims to bridge the gaps between plants, people, microbes, and the environment through interdisciplinary research. The center supports a variety of STEM outreach efforts, including a 10-week summer program that allows undergraduate students to conduct research at the intersection of plant biology, engineering, and computer science. CROPPS also collaborates with the Office of Inclusive Excellence, Cornell Engineering's CURIE and CATALYST Programs, which are outreach initiatives for rising high school juniors and seniors. As part of a week-long on-campus experience, students engage with CROPPS researchers and explore the concept of communicating with plants. The program begins by introducing biological and engineering principles, which students apply in hands-on experiments, such as using sensors to measure soil moisture. This educational model closely parallels our approach to programmable plant systems and demonstrates the impact of emerging technologies integrated learning experience.

Summer 2024, CURIE Program students using sensors to measure soil moisture at the Cornell Orchards.

Although several of these organizations are working at a much larger scale compared to our team, there are several key takeaways for us from both these organizations and the previous IoT in STEM teams.

One of the main takeaways is that a more hands-on approach is more likely to yield positive results. Because IoT consists heavily of circuits, students need to learn how all these parts fit together by putting them together themselves as much as

[1] https://locorobo.co/

[2]https://www.ruggedtelemetry.com/solutions/iot-for-stem#:~:text=Rugged%20Telemetry%20is%20proud%20to,evolving%20ecosystem%20of%20IoT%20technology

possible. As the current team taking on this project, we want to use the hands-on approach as much as possible. We intend to take the approach of presenting the final product to them in its basic components, similar to how you get your LEGO tower in the form of individual blocks and have to put it together. As they put these blocks together (sensors, boards, cables, code), they will better be able to understand how one thing fits together with and informs the other.

Additionally, one of our observations is that it is important for IoT in the classroom to remain open ended. When we introduce a product, device, or system that uses IoT to students, it is more important for us to invoke their creativity by simply presenting them with a finished product (which would be no different from them buying a phone, for example). It is important for them to understand how the product works so that they are exposed to the possibility of making that product better or building on it. That is one of the key aspects that we recognize is part of the works that have inspired us to build our design.

Another key takeaway for us is emphasizing the relevance of IoT in the real world to the students. As engineering students, we are fully convinced that IoT has practical applications, which is why we have pursued a career in engineering, but it may not be as obvious for high school students. To give them a similar level of understanding of IoT applications, it is a good idea to infuse IoT into something they already know about or are already learning about. The previous team did this, and LocoRobo and Rugged Telemetry used a similar approach. Like the previous team, we chose a course that we could use to show the usefulness of IoT systems, which is what led us to our current idea. By connecting IoT with something students already know, we are showing them in a practical way how IoT can be used, and in this way, they are more likely to develop an interest in using IoT for different things. This will also encourage them to explore their imaginations and think about what IoT systems could be useful for in the real world.

Project Scope

Vision

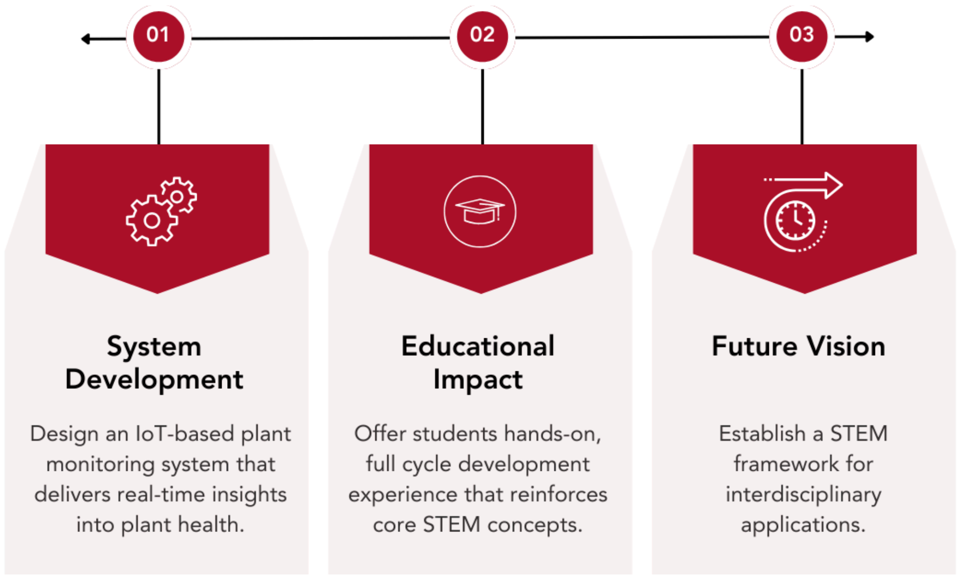
The vision of this project is to design and demonstrate a framework for a STEM education model that will serve as the template for future projects applied to other disciplines in high school classrooms. This project empowers students to address real-world challenges in environmental sustainability through the integration of Internet of Things (IOT) Technology. By developing an IoT-based Plant Monitoring System, students will engage in hands-on learning experiences involving system design, data collection, and analysis to better understand and optimize plant growth conditions.

This design project was shaped through collaboration with our community partner, Shelley Walker, an environmental botany teacher at Geneva High School. Her curriculum covers topics such as plant anatomy, ecological interactions, photosynthesis, soil science, conservation methods, and sustainability. Students are already introduced to tools that measure pH, oxygen, nitrates, phosphorus, and soil moisture, The Plant Monitoring system builds on that foundation by incorporating real-time sensing and digital data collection.

Our goal is to provide students with an enhanced, technology-driven method for exploring plant sustainability, an initiative that improves accuracy and accessibility through the use of sensors. This project fosters a deeper understanding of environmental science and IoT systems while enhancing critical STEM skills, preparing Geneva High School students for academic and professional success in the technology-driven world.

Project Objectives

The project is divided into three main objectives: System Development, Educational Impact, and Future Vision.


01
02
03
System Development
Design an IoT-based plant monitoring system that delivers real-time insights into plant health.
Educational Impact
Offer students hands-on, full cycle development experience that reinforces core STEM concepts.
Future Vision
Establish a STEM framework for interdisciplinary applications.

System Development

- Create a hands-on system that teaches mechanical principles through physical assembly.
- Integrate sensors to collect data, including soil moisture levels, temperature, and humidity.
- Develop a data pipeline and provide an analytics platform (such as a website) to visualize environmental trends and system performance.

Educational Impact

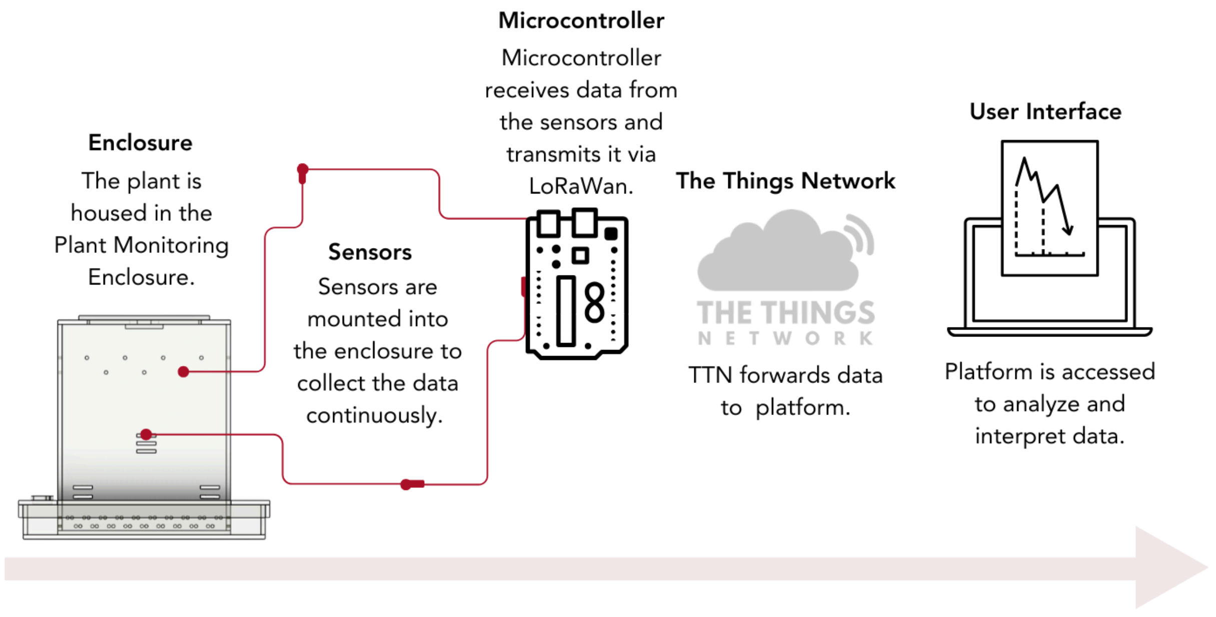
- Engage students in a full-cycle development process, from design and assembly to testing and analysis.
- Reinforce core STEM concepts through experiential learning in electronics, programming, and environmental science.
- Provide a meaningful application of environmental science topics from the classroom curriculum.
- Design an assessment tool to evaluate both student learning outcomes and system effectiveness.

Future Vision

- Establish a replicable STEM framework that can be adapted to other disciplines and subjects.
- Consider curriculum alignment and long-term impacts to strengthen community partnerships.

IoT Architecture of the Plant Monitoring System

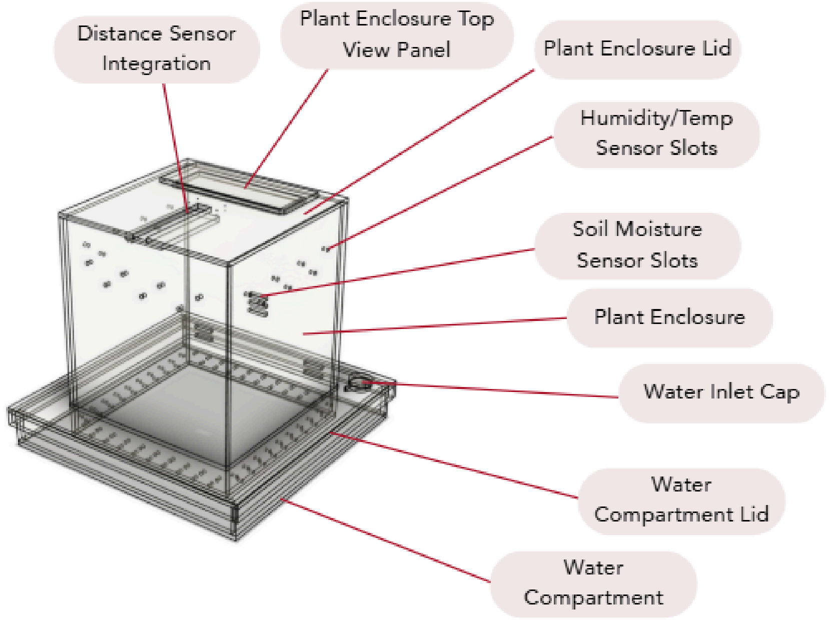
The Plant monitoring system consists of an integrated enclosure, hardware, and software designed to collect, transfer, and visualize data in real time.


Microcontroller
Microcontroller receives data from the sensors and transmits it via LoRaWan.
Enclosure
The plant is housed in the Plant Monitoring Enclosure.
Sensors
Sensors are mounted into the enclosure to collect the data continuously.
The Things Network
THE THINGS NETWORK
TTN forwards data to platform.
User Interface
Platform is accessed to analyze and interpret data.

Technological Development

Enclosure Design

Overview | The enclosure consists of three modular components: a black lid, an acrylic plant enclosure, and an acrylic water box. We selected wood for the lid to prevent external light sources while using transparent acrylic for the other components to enable students to observe root development and monitor water levels visually. Details of the assembly process are detailed in Appendix VIII.


Distance Sensor Integration
Plant Enclosure Top View Panel
Plant Enclosure Lid
Humidity/Temp Sensor Slots
Soil Moisture Sensor Slots
Plant Enclosure
Water Inlet Cap
Water Compartment Lid
Water Compartment

Lid: Light Control | The lid serves two purposes: light integration and sensor integration. Its primary function is to create a controlled environment where the mounted grow lights (operational conditions: high/low modes by students) serve as the sole light source. A center slot holds the wiring for an optional distance sensor, which we may substitute with a ruler to perhaps compare digital versus analog measurement methods. Also, we are unsure of how efficient a distance sensor will be since plants don't grow in a vertical, straight manner. This design intentionally introduces fundamental scientific principles. Students will explore how plants convert light energy to chemical energy, prompting critical questions about optimal light conditions for plant health.

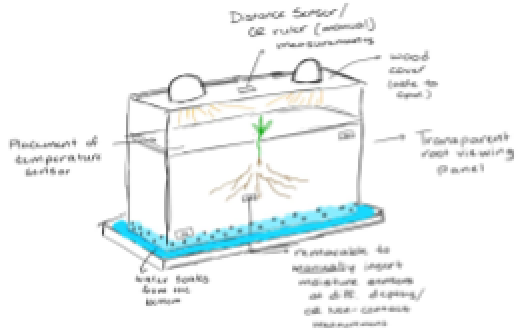
Plant Enclosure: Growth Environments and Sensor Access Ports | The plant enclosure features three key elements. First, a reverse watering system with rows of small holes at the base that utilize capillary action to water the soil from the bottom while protecting the top-mounted sensors from water damage. Second, a back panel with multiple holes for the placement of the humidity/temperature sensor wires; the unused ports will be sealed with corks. Third, a front panel featuring soil sensor slots and tape can be used to cover the unused openings. These elements demonstrate the effectiveness of mechanical design, how it impacts plant health through watering efficiency and data accuracy via sensor placement. Students will experiment with how sensor positioning affects data accuracy.

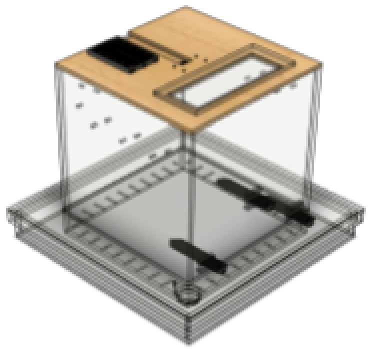
Water Box: Structural Support and Reverse Watering System | The water box serves as the structural foundation and the safety component of the design. It is made out of transparent acrylic to allow students to monitor water levels visually. It has a removable lid that features a water inlet to prevent splashing and protect the electrical components. It's designed with four internal vertical supports that securely position the plant enclosure.

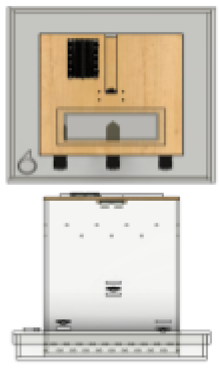
Hands-on Assembly Process | To begin assembly, place the water compartment on a flat surface. Position the plant enclosure into the water compartment, then place the removable water compartment lid, and finally secure the plant enclosure with the lid.

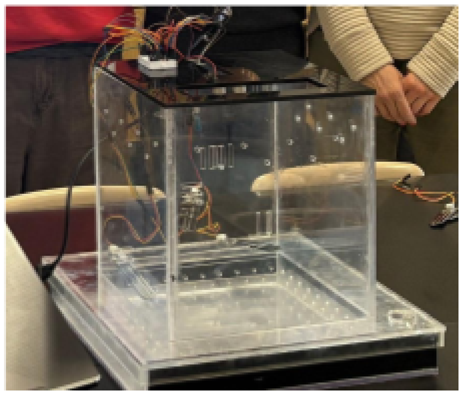
Sensor Integration | The enclosure includes pre-drilled screw holes on the lid to attach a distance sensor for measuring plant height. Designated slots are included to place soil moisture sensors. Wires for the humidity and temperature sensors can be routed through the circular ports, which can be sealed with corks if unused.

The figure illustrates the progression from the initial enclosure design sketch, to the Fusion 360 model with hardware in place, and finally to the completed functional physical prototype installed in Ms. Walker's environmental botany classroom.


Distance Sensor/ OR ruler (manual) measurements
Wood cover (safe to open)
Placement of temperature sensor
Transparent root viewing panel
Water soaks from the bottom
removable to manually insert moisture sensors at diff. depths/ OR non-contact measurements







Sensor Selection and Justification

The system consists of three soil moisture sensors, one temperature and humidity sensor, and an ultrasonic distance sensor. These sensors were chosen based on the learning objectives that we want to achieve, and to continue this design and expand the analysis for future years.

1) *Soil Moisture Sensor* | These are capacitive sensors that will detect the water content in the soil. These sensors use dielectric theory to give an analog data reading that represents the moisture level. This is to determine whether or not a plant has enough water to keep on growing. It has an operating voltage similar to the ones provided by the Featherboard (3.3V and 5V), and it has a pin where the data information can be received. Due to the slow rate of change, this data can be acquired every hour and does not require any special considerations.
2) *Temperature and Humidity Sensor* | Similar to the soil moisture sensor, this sensor has three pins, a positive and negative lead, and a data cable that will return the temperature and humidity information. This information is important to ensure proper living conditions for the plant to grow similar to its native environment. Since this data is more variable and dependent on external factors (such as ambient temperature), this data should be collected more frequently, but it does not require any special considerations.
3) *Ultrasonic Distance Sensor* | The ultrasonic distance sensor has four pins: a positive and negative lead, a trigger pin, and an echo pin. These two pins are special to the distance sensor as the trigger sends out the ultrasonic wave, and the echo pin relates the speed of sound to the distance. This sensor will gather information about how the plant is growing. Due to the slow growth of the plant, this data can be collected once a day, and the data does not require any special considerations. This ultrasonic sensor is suitable for the scope of our design because our suggested plant to be used can typically grow without supports.

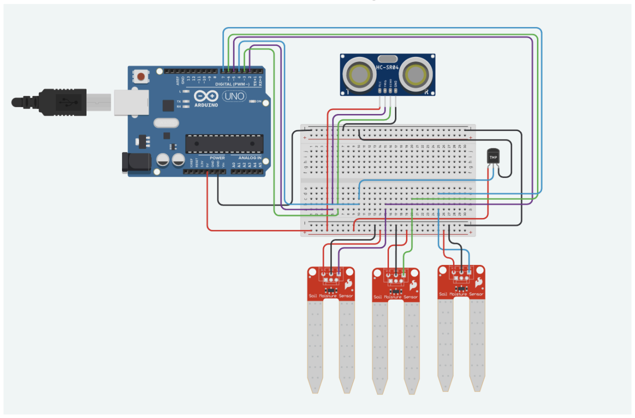
Wireless Communication and System Integration

Much of our code makes use of classes whose methods simply need to be called in the final implementation. Each sensor has a class and designated methods; data obtained from the sensors will be sent to TTN in a similar way (a TTN class with methods to send the data). This simplifies the coding aspect of the project enough that even students with little or no coding experience will be able to follow the general logic of the program.

In our prototype, we have designated slots for each of the sensors we will be using on the sides and top of the box. The sensors are placed so that they can easily record the targeted parameters (soil temperature/moisture, humidity, distance).
Our code allows the students to send the sensor data to TTN via LoRaWAN through straightforward implementations. Our code assigns a value from the sensor to the corresponding attribute in the TTN class, adds it to the data packet, and sends it to TTN. Since we are recording values that can potentially remain relatively constant for significant periods, we need to be able to control how often we send data to TTN. That makes our current method of sending data more suitable since it allows us to modify how often we send data. This modification also saves power that would be used to send redundant information to TTN.

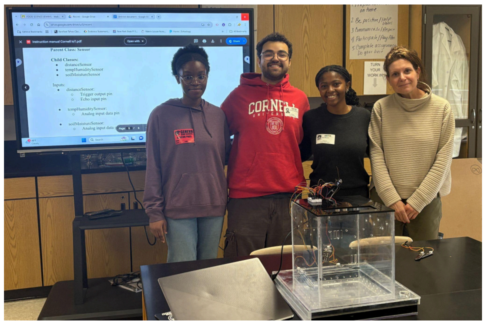
This data is useful to the educator we are working with because it provides useful information on the conditions under which the plant of interest can thrive best. This provides a practical, hands-on reinforcement to the information that is provided in theory in textbooks, for instance.

The circuit connections we used for testing are as shown in the figure below.



Presentation to Students at Geneva High School

We, as a team, were able to present this project to the Geneva High School students on May 9, 2025, with our enclosure. Our development process required the construction of an instruction manual that would be utilized by the students for them to have an understanding of the systems at hand, as well as a general overview of the project. During our visit, we provided the classroom with hard copies of the manual, as well as using their projector to go over the entirety of the manual as our session with the students progressed. Together, we went through working principles of the sensors, understanding of Arduino and breadboards, understanding circuit diagrams, and gaining familiarity with IoT and TTN. Through the presentation and interaction the students were able to defer to the handout if they had questions, as well as enquire about the project during the presentation. The Instruction Manual presented to the students can be found in Appendix XIII.



Field Deployment

Throughout our presentation, students were able to gain hands-on experience through connecting the sensors and getting a representation of what the circuit diagram actually looks like. Though we were only able to present for 40 minutes, we were able to go through all of the basics as well as all of the hardware connections in order for them to be able to fill the enclosure the following week.

Before leaving Geneva, we uploaded all of our final code to the featherboard so that once the system is connected to power it will be able to run without our physical presence. We left the enclosure as well as extra sensors in case there is a need for replacement or if there is some time of hardware issue. With the instruction manual, the students will be able to reconnect the sensors if needed. As of now, we have communicated with Mrs. Walker and the students have filled the enclosure and the system has been connected to power.

Conclusions & Next Steps

Our plan for this project was to develop the design, create working code, and have the product deployed to the school so that the students could actively interact with the incoming data. At the conclusion of the project, we have managed to get the product to the students and explain to them what it does and how it works. However, we have not developed a reliable way to observe and analyze the incoming data. Additionally, we have not been able to extend the reach of our project to more students taking the class. Other students may get to see the product, but we were only able to have a hands-on learning session with a few of the students.

In the future, we hope to have a reliable system for receiving data and displaying it in a meaningful way. The idea is to produce something from the IoT project that will be relevant to what the students are currently learning. By developing a way to display the data and plot it or follow trends, IoT can become a useful tool in the high school curriculum, beginning with this specific project and class (AP Botany). Currently, we can only send the data into a file or table locally. Developing a simple website would make for better ease of access for the students.

Furthermore, it would be beneficial to extend the reach of the project by making more trips to the school or using an asynchronous method (some sessions online and some in person) for sharing the tutorials and updates. This would mean more total sessions with the students and possibly a wider reach.

Other additions include adding a lamp inside the box (on the inner part of the lid) which would provide extra warmth if needed. Power considerations can also be made, i.e figuring out a way to have the box continue running even when it is not plugged into the wall (backup power). Changing the frequency of data transmission is another modification that would reserve power. In addition to that, our box will currently lose its collected data every time it is disconnected from power. Incorporating FRAM storage into the circuitry and code would prevent data loss when the box is disconnected from power. This would make the collected data usable for longer periods of time.

Although our intention was to start meeting the students as early as possible, we were slightly held back due to slowed communication. Because this project heavily relies on communication with teachers, planning can often be difficult since teaching is one of the busiest occupations and the teachers in contact simply may not find the time to respond. Other than that, gauging the students' proficiency in STEM related topics can be tricky, and it is an important factor when deciding what parts of the project will be left for the students to do.

The plant box is a promising potential tool; it shows promise for products of its kind. Incorporating IoT into classes in the way that we did with the plant box is a good way to naturally introduce IoT to students who are not very familiar with STEM and IoT. For large scale use, a smaller version of the plant box that uses portable power and monitors one or two parameters for a small plant would be a good starting point for introducing IoT in biology, science or botany classes.

Reference

IEEE Standards Association. IEEE Standards Activities in the Internet of Things (IoT). IEEE, 2018. https://standards.ieee.org/wp-content/uploads/import/documents/other/iot.pdf.

Wheeler/Provided, Credit: Simon, et al. "CROPPS Welcomes 2024 REU Class." Cornell Chronicle, 5 June 2024, news.cornell.edu/stories/2024/06/cropps-welcomes-2024-reu-class.

Sinha, Satyajit. "State of IOT 2024: Number of Connected IOT Devices Growing 13% to 18.8 Billion Globally." *IoT Analytics*, 10 Feb. 2025, iot-analytics.com/number-connected-iot-devices/#:~:text=State%20of%20IoT%202023:%20Number,Sinha%20on%20May%2024%2C%202023.

"Why Is Stem Important in Early Childhood Education? Understanding Child Development and Learning: American University." *School of Education Online*, 15 Apr. 2024, soeonline.american.edu/blog/stem-in-early-childhood-education/#:~:text=Lifelong%20Benefits%20of%20Early%20STEM%20Education&text=For%20example%2C%20a%20study%20by,kindergarten%20to%20fifth%20grade%20students.

Afzal Badshah, Anwar Ghani, Ali Daud, Ateeqa Jalal, Muhammad Bilal, and Jon Crowcroft. 2023. Towards Smart Education through Internet of Things: A Survey. ACM Comput. Surv. 56, 2, Article 26 (February 2024), 33 pages. https://doi.org/10.1145/3610401

Appendices

Appendix I: Sensor Class Header File

```
#ifndef SENSOR_H
#define SENSOR_H
#include <DHT11.h>

// Sensor base class
class Sensor {
  public:
    Sensor(); // Constructor declaration

    virtual void sensorType(); //base virtual method indicating the type of sensor in question
    virtual void begin(); // Initialize sensor for data collection
    void ToSerial();
    void ToString();
};

// Temperature Humidity Sensor class
class tempHumiditySensor : public Sensor {
  public:
    tempHumiditySensor(int analogPin1): tempSensor(analogPin1){
      _analogPin1 = analogPin1;
    };

    virtual void sensorType() override; // Override base method
    virtual void begin() override; // Override base method
    float readTempData(); // Enclosure Temp Sensor data
    float readHumidityData(); // Enclosure Humidity Sensor Data

  private:
    int _analogPin1;
    DHT11 tempSensor;
};

// Distance Sensor class
class distanceSensor : public Sensor {
  public:
    distanceSensor(int triggerPin, int echoPin); //distance sensor constructor declaration
    virtual void sensorType() override; // Override base method
    virtual void begin() override; // Override base method
    float readData(); // Plant height sensor data

  private:
    int _triggerPin;
    int _echoPin;
    float boxHeight = 276.2239;
    float soilHeight = 120;
};

// Soil Moisture Sensor class
class soilMoistureSensor : public Sensor {
  public:
    soilMoistureSensor(int analogPin2);  //soil moisture sensor constructor declaration

    virtual void sensorType() override; // Override base method
    virtual void begin() override; // Override base method
    float readData(); // Soil moisture content sensor data

  private:
    int _analogPin2;

};

#endif
```

Appendix II: Sensor Class cpp file

```
#include "Arduino.h"
#include "Sensor.h"

// Constructor definition for a generic sensor
Sensor::Sensor() {

}

//Constructor definition for distance sensor

// Base class method implementation
void Sensor::sensorType() {
  Serial.println("This is not an active sensor\n");
}

void Sensor::begin() {
    Serial.println("There is no beginning protocol as this sensor is not active.\n");
}

//Temperature Humidity Sensor Class implementation
void tempHumiditySensor::sensorType() {
  Serial.println("Temperature/Humidity Sensor\n");
}

// Sensor specific beginning protocol
void tempHumiditySensor::begin() {
  pinMode(_analogPin1, INPUT);
  Serial.println("There is no initialization required for the Temperature Humidity Sensor!\n");
}

// Temperature Specific Data method
float tempHumiditySensor::readTempData() {
  return (float) tempSensor.readTemperature();
}

// Humidity Specific Data method
float tempHumiditySensor::readHumidityData() {
  return (float) tempSensor.readHumidity();
}
```

```
distanceSensor::distanceSensor(int trigger, int echo) {
  _triggerPin = trigger;
  _echoPin = echo;
}
//Distance Sensor Class implementation
void distanceSensor::sensorType() {
    Serial.println("Distance Sensor\n");
}

// Sensor specific beginning protocol
void distanceSensor::begin() {
  pinMode(_triggerPin, OUTPUT);
  pinMode(_echoPin, INPUT);
  digitalWrite(_triggerPin, LOW);

  Serial.println("The distance sensor has been initialized\n");
}

// Distance Data method
float distanceSensor::readData() {
  digitalWrite(_triggerPin, LOW);
  delayMicroseconds(5);
  digitalWrite(_triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(_triggerPin, LOW);

  while(digitalRead(_echoPin) == LOW); //The code will pause until a signal from the echo pin is
recieved

  int firstPulse = micros(); //Time of the first pulse recieved from the echo pin

  while(digitalRead(_echoPin) == HIGH); //Pause until the signal stops being recieved

  int duration = micros() - firstPulse; //Difference in time

  float distance = (duration / 2) * 0.332; //wave traveled the distance twice and the speed of
sound is 0.343 mm/us but we need to account for the pulse length which is 10 us

  float plantHeight = boxHeight - distance - soilHeight;

  return plantHeight/10;
}
```

```
soilMoistureSensor::soilMoistureSensor(int analogPin2) {
  _analogPin2 = analogPin2;
}

//Distance Sensor Class implementation
void soilMoistureSensor::sensorType() {
    Serial.println("Soil Moisture Sensor\n");
}

// Sensor specific beginning protocol
void soilMoistureSensor::begin() {
    Serial.println("There is no initialization required for the Soil Moisture Sensor!\n");
}

// Moisture Data method
float soilMoistureSensor::readData() {
  int maxMoisture = 330 ;
  int minMoisture = 860 ;
  int samples = 128;
  int interval = 10;
  int totalSum = 0;

  for (int i = 0; i < samples; i++){
    delay(interval);
    totalSum += analogRead(_analogPin2);
  }

  float value = totalSum/samples;

  float percentage = (minMoisture - value)/(minMoisture-maxMoisture) * 100;
  if (abs(percentage) <= 2) {
    percentage = 0;
  }

  return percentage;

}
```

Appendix III: ttnData Class Header File

```
#ifdef COMPILE_REGRESSION_TEST
#define FILLMEIN 0
#else
#define FILLMEIN //(#Don't edit this stuff. Fill in the appropriate FILLMEIN values.)
#warning "You must fill in your keys with the right values from the TTN control panel"
#endif

#include <Arduino_LoRaWAN_ttn.h>
#include <lmic.h>
#include <hal/hal.h>
#include "keys.h"

class cMyLoRaWAN : public Arduino_LoRaWAN_ttn {
public:
    cMyLoRaWAN() {};

protected:
    // you'll need to provide implementations for each of the following.
    virtual bool GetOtaaProvisioningInfo(Arduino_LoRaWAN::OtaaProvisioningInfo*) override;
    virtual void NetSaveSessionInfo(const SessionInfo &Info, const uint8_t *pExtraInfo, size_t
nExtraInfo) override;
    virtual void NetSaveSessionState(const SessionState &State) override;
    virtual bool NetGetSessionState(SessionState &State) override;
    virtual bool GetAbpProvisioningInfo(Arduino_LoRaWAN::AbpProvisioningInfo*) override;
};

class ttnData {
  public:
    ttnData();

    void begin();
    void sendPacket();
    float temp;
    float humidity;
    float distance;
    float moisture[3];

    cMyLoRaWAN myLoRaWAN;

};
```

Appendix IV: ttnData Class cpp file

```
#include "SensorToTTN.h"
#include "Arduino.h"
#include <Arduino_LoRaWAN_ttn.h>
#include "keys.h"


// The pinmap. This form is convenient if the LMIC library
// doesn't support your board and you don't want to add the
// configuration to the library (perhaps you're just testing).
// This pinmap matches the FeatherM0 LoRa. See the arduino-lmic
// docs for more info on how to set this up.
const cMyLoRaWAN::lmic_pinmap myPinMap = {
     .nss = 8,
     .rxtx = cMyLoRaWAN::lmic_pinmap::LMIC_UNUSED_PIN,
     .rst = 4,
     .dio = { 3, 6, cMyLoRaWAN::lmic_pinmap::LMIC_UNUSED_PIN },
     .rxtx_rx_active = 0,
     .rssi_cal = 0,
     .spi_freq = 8000000,
};

void myStatusCallback(void * data, bool success){
  if(success)
    Serial.println("Succeeded!");
  else
    Serial.println("Failed!");

}

struct __attribute__((__packed__)) buildPacket{
  float temp1;
  float humidity1;
  float distance1;
  float moisture1[3];
};

ttnData::ttnData() {
  temp = 0;
  humidity = 0;
  distance = 0;
  moisture[0] = 0;
  moisture[1] = 0;
  moisture[2] = 0;
  cMyLoRaWAN myLoRaWAN {};
};

void ttnData::begin() {
    Serial.begin(115200);

    uint64_t lt = millis();
    while(!Serial && millis() - lt < 10000);

    myLoRaWAN.begin(myPinMap);
    lt = millis();
    Serial.println("Serial begin");

    if(myLoRaWAN.IsProvisioned())
      Serial.println("Provisioned for something");
    else
      Serial.println("Not provisioned.");

  return;
}

void ttnData::sendPacket() {
    buildPacket data;
    data.temp1 = temp;
    data.humidity1 = humidity;
    data.distance1 = distance;
    data.moisture1[0] = moisture[0];
    data.moisture1[1] = moisture[1];
    data.moisture1[2] = moisture[2];

    uint32_t bufferLength = sizeof(data);
    static uint8_t messageBuffer[sizeof(data)];

    memcpy(messageBuffer, (uint8_t *) &data, 20);
    myLoRaWAN.SendBuffer(messageBuffer, bufferLength, myStatusCallback, NULL, false, 1);

    return;
}
```

```
bool cMyLoRaWAN::GetOtaaProvisioningInfo(
    OtaaProvisioningInfo *pInfo
    ) {
      if (pInfo){
        memcpy_P(pInfo->AppEUI, APPEUI, 8);
        memcpy_P(pInfo->DevEUI, DEVEUI, 8);
        memcpy_P(pInfo->AppKey, APPKEY, 16);
      }
    return true;
}

void cMyLoRaWAN::NetSaveSessionInfo(
    const SessionInfo &Info,
    const uint8_t *pExtraInfo,
    size_t nExtraInfo
    ) {
    // save Info somewhere.
}

void cMyLoRaWAN::NetSaveSessionState(const SessionState &State) {
    // save State somwwhere. Note that it's often the same;
    // often only the frame counters change.
}

bool cMyLoRaWAN::NetGetSessionState(SessionState &State) {
    // either fetch SessionState from somewhere and return true or...
    return false;
}

bool cMyLoRaWAN::GetAbpProvisioningInfo(Arduino_LoRaWAN::AbpProvisioningInfo* Info){
  //either get ABP provisioning info from somewhere and return true or...
  return false;
}
```

Appendix V: Main Script

```
#include "Arduino.h"
#include "Sensor.h"
#include "SensorToTTN.h"

ttnData plantBox;

uint64_t lastTime = 0;

distanceSensor D1(A0, A1);
tempHumiditySensor T1(A2);
soilMoistureSensor S1(A3);
soilMoistureSensor S2(A4);
soilMoistureSensor S3(A5);

void setup() {
  plantBox.begin();
  D1.begin();
  T1.begin();
  S1.begin();
  S2.begin();

}

void loop() {
    plantBox.myLoRaWAN.loop();
    if (millis() - lastTime > 21600000){ //equivalent to 6 hours
      plantBox.distance = D1.readData();
      plantBox.humidity = T1.readHumidityData();
      plantBox.temp = T1.readTempData();
      plantBox.moisture[0] = S1.readData();
      plantBox.moisture[1] = S2.readData();
      plantBox.moisture[2] = S3.readData();
      plantBox.sendPacket();
      lastTime = millis();
    }
}
```

Appendix VI: TTN Uplink Payload Formatter (JavaScript)

```
//All the code here was acquired from the Lab3 V2 handout

function build_uint_from_bytes(bytes) {
    var built_up_uint = 0; // Initialize the result as 0
    for (var i = 0; i < bytes.length; i++) {
      // Add the value of each byte shifted to its        appropriate position
      // '<<' shifts the byte value by (i * 8) bits to its place in the integer
      built_up_uint = built_up_uint + (bytes[i] << i * 8);
    }
   return built_up_uint; // Return the constructed unsigned integer
 }

 // Function to build a signed integer from an array of bytes
 // This function interprets the bytes as a signed integer in little-endian format

 function build_int_from_bytes(bytes) {
   var built_up_int = 0; // Initialize the result as 0
   var cap = bytes.length - 1; // Determine the index of the most significant byte

   // Process all bytes except the most significant byte
   for (var i = 0; i < cap; i++) {
     // Add the value of each byte shifted to its appropriate position
     built_up_int = built_up_int + (bytes[i] << i * 8);
   }

   // Handle the most significant byte separately to account for its signedness
   var lastByte = bytes[cap]; // Get the most significant byte

   if (lastByte & 0b10000000) {
     // Check if the most significant bit (MSB) is set, indicating a negative value
     lastByte = lastByte - 256;
     // Convert the byte to a signed value (e.g., 0xFF becomes -1)
     built_up_int = built_up_int + (lastByte << cap * 8);
     // Add the signed value to the total, shifted to its proper position
   }

   return built_up_int; // Return the constructed signed integer
 }

 function stringFromBytes(bytes){
  return String.fromCharCode(...bytes);
}

function floatFromBytes(bytes) {
    // JavaScript bitwise operators yield a 32 bit integer, not a float.
    // Assume LSB (least significant byte first).
    var bits = bytes[3]<<24 | bytes[2]<<16 | bytes[1]<<8 | bytes[0];
    // >>> is unsigned right shift. Zeros are shifted in at all times from the left
    var sign = (bits>>>31 === 0) ? 1.0 : -1.0;
    var e = bits>>>23 & 0xff;
    // Can code explicit leading 0 with 0 exponent. Otherwise, there is a
    // leading 1 implied.
    // Since we have constructed our significand/mantissa like we constructed
    // our integers, it is a factor of 2^23 too large (decimal point all the way)
    // right.
    // So, we correct that when we build the final number.
    var m = (e === 0) ? (bits & 0x7fffff)<<1 : (bits & 0x7fffff) | 0x800000;
    var f = sign * m * Math.pow(2, e - 127-23);
    return f;
  }

function decodeUplink(input) {
  return {
    data: {
      boxTemperature : (floatFromBytes(input.bytes.slice(0,4))),
      boxHumidity : (floatFromBytes(input.bytes.slice(4,8))),
      plantHeight : (floatFromBytes(input.bytes.slice(8,12))),
      moisture1 : (floatFromBytes(input.bytes.slice(12,16))),
      moisture2 : (floatFromBytes(input.bytes.slice(16,20))),
      moisture3 : (floatFromBytes(input.bytes.slice(20,24))),
    },
    warnings: [],
    errors: []
  };
}
```

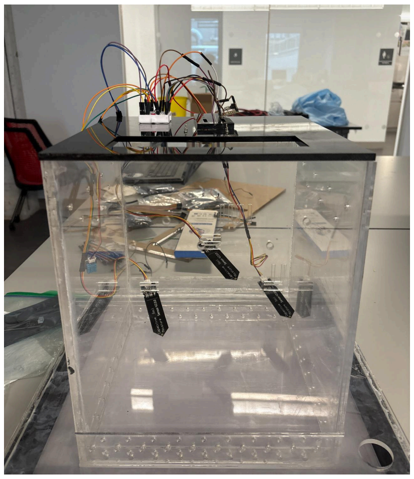
Appendix VII: Data Analysis Code (Update)

```
# app.py
from flask import Flask, render_template, jsonify
import paho.mqtt.client as mqtt
import json
import time
import threading
from datetime import datetime
from dateutil.parser import parse
import pytz
import collections

app = Flask(__name__)

# Configuration
TTN_APP_ID = "plant-box@ttn"
TTN_API_KEY =
"NNSXS.SUFBQH4J3UNWANYVWQRGM4HQ6RRBKWUOJ5OBEKQ.DTGXKCCN7NG66NQY
KM3VGUKY7JKQDFP3JC6ZBBKTCLSVQUZJGFQQ"
BROKER = "nam1.cloud.thethings.network"
PORT = 1883
TOPIC = f"v3/{TTN_APP_ID}/devices/+/up"

MAX_RECORDS = 100
data_lock = threading.Lock()
sensor_data = collections.deque(maxlen=MAX_RECORDS)

# MQTT Client
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to TTN MQTT broker")
        client.subscribe(TOPIC)
    else:
        print(f"Failed to connect to MQTT broker with code {rc}")

def on_message(client, userdata, message):
    try:
        # Parse the message
        payload = message.payload.decode()
        js = json.loads(payload)

        # Extract timestamp and convert to Eastern Time
        ts = parse(js["received_at"]).astimezone(pytz.timezone("US/Eastern"))

        # Extract the decoded payload
        decoded = js["uplink_message"]["decoded_payload"]


        record = {
            "time": ts.strftime("%Y-%m-%d %H:%M:%S"),
            "device_id": js["end_device_ids"]["device_id"],
            # Add all the sensor fields you need
            "temperature": decoded.get("temp"),
            # Add any other fields from your data
            "raw_payload": payload  # Store the raw payload for debugging
        }

        with data_lock:
            sensor_data.appendleft(record)

        print(f"New data received from {record['device_id']}")

    except Exception as e:
        print(f"Error processing message: {e}")

# Start MQTT client in a background thread
def mqtt_client_thread():
    client = mqtt.Client("PythonWebDashboard")
    client.username_pw_set(TTN_APP_ID, password=TTN_API_KEY)
    client.on_connect = on_connect
    client.on_message = on_message

    while True:
        try:
            client.connect(BROKER, PORT, 60)
            client.loop_forever()
        except Exception as e:
            print(f"MQTT connection error: {e}")
            print("Reconnecting in 10 seconds...")
            time.sleep(10)

# Start the MQTT client thread when the app starts
mqtt_thread = threading.Thread(target=mqtt_client_thread, daemon=True)
mqtt_thread.start()
```

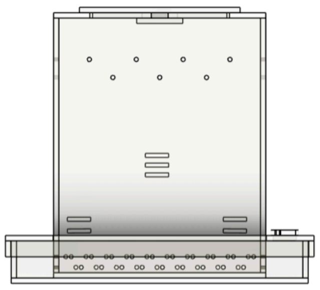
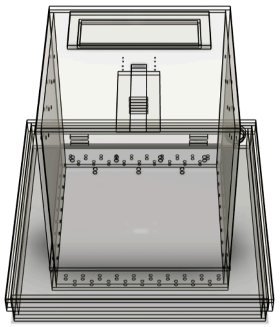
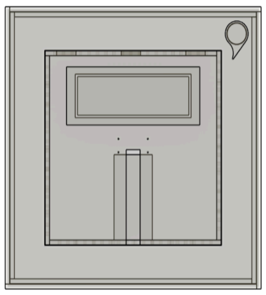
```
# Flask routes
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/api/data')
def get_data():
    with data_lock:
        # Convert deque to list for JSON serialization
        data_list = list(sensor_data)
    return jsonify(data_list)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Appendix VIII: Plant Enclosure Design and CAD

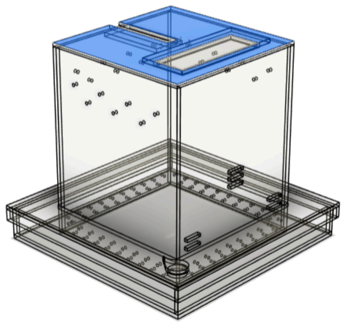
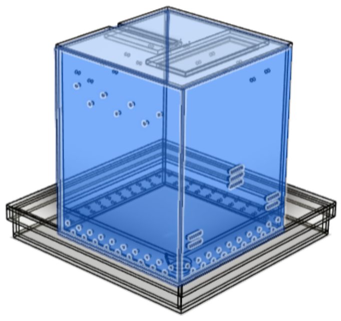
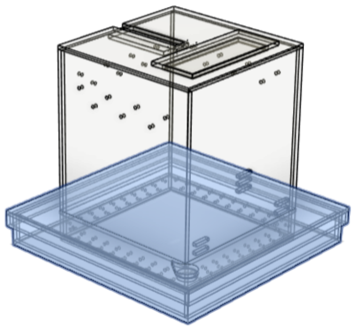
Capabilities: easy assembly, sensor integration, water damage proof, transparent view of plant
Material: ¼"Thick Black and Clear Acrylic
Fabrication Method: DXF files were created from a Fusion 360 CAD file to be Laser Cut
Total Surface Area: 1136.473 in
Overall Dimensions: 14.5 in x 14.5 in x 12.5 in
Assembly Method: All components were bonded using acrylic cement
Assembly Duration: 3 hours to put together, 90 hours to reach the complete bond strength
Clearance (when assembling in Fusion 360): 0.001 m



Perspectives

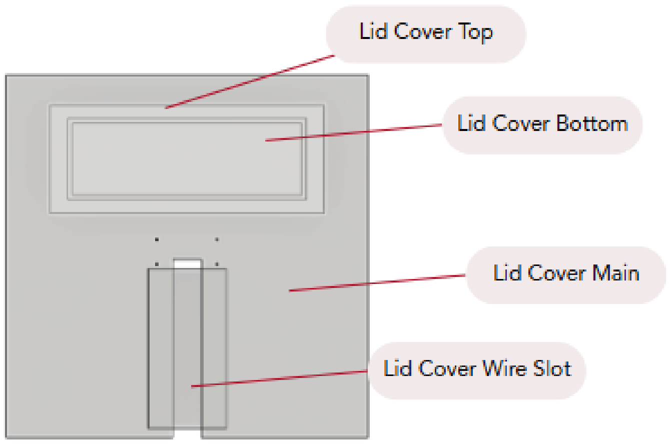
Front View	Isometric View	Top View
		

Full Assembly

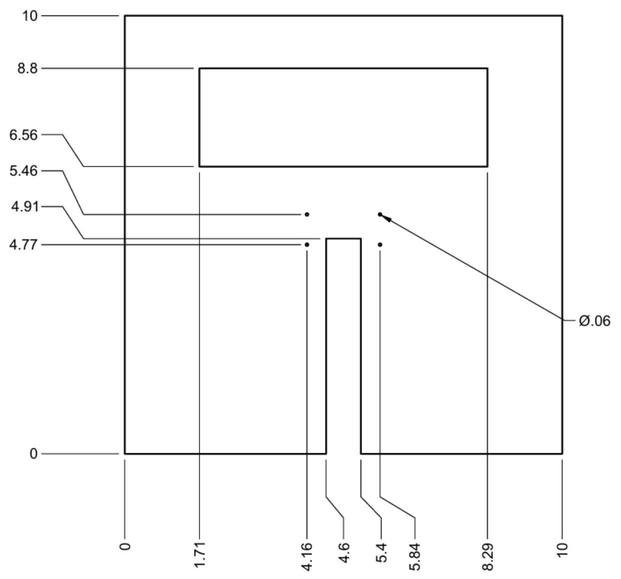
Lid	Plant Enclosure	Water Compartment
		

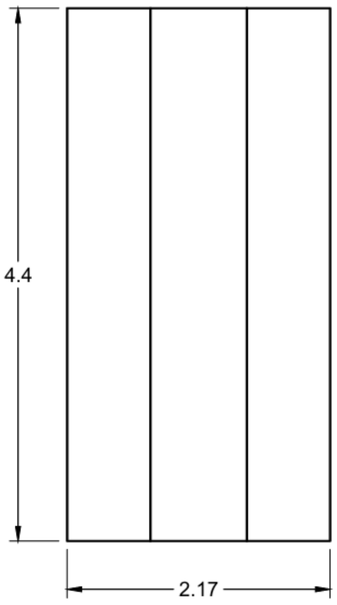
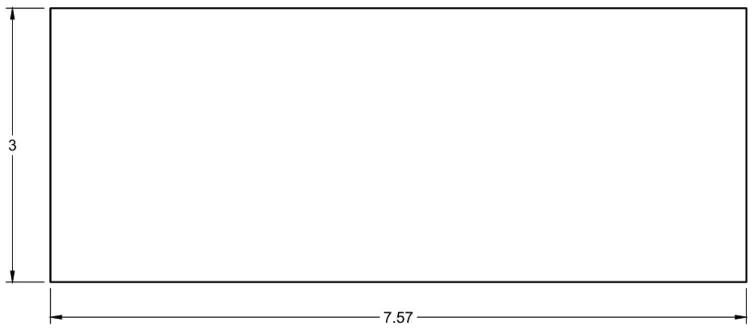
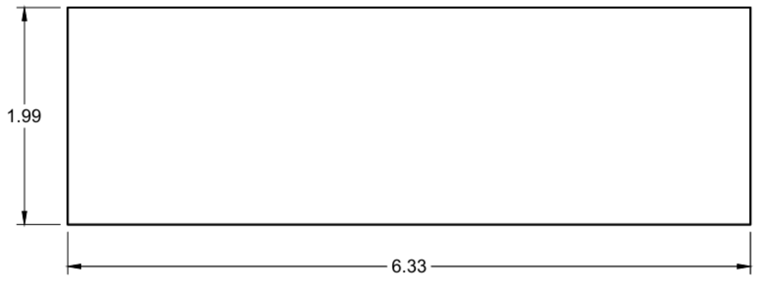
Measurements

Lid

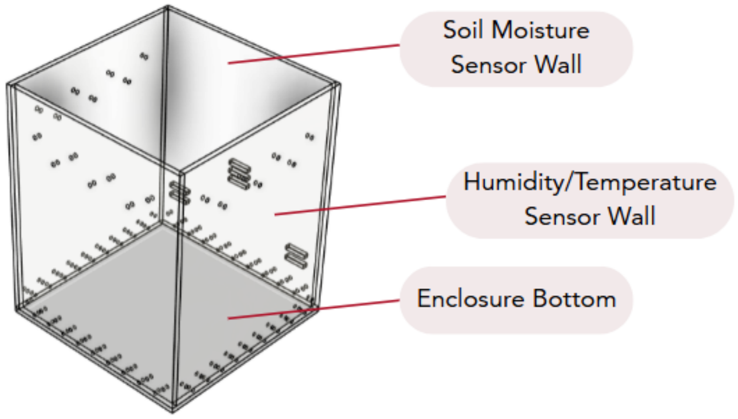


The Lid Cover Top is attached to the Lid Cover Bottom to form a single removable viewing panel. This assembled piece is not permanently fixed to the enclosure, allowing students to easily remove it for inspection, sensor adjustment, or light replacement. The Lid Cover Wire Slot is attached to the Lid Cover Main along the centerline, positioned precisely 0.359 inches above the bottom edge. This slot allows for the routing of sensor wiring (such as the distance sensor) while keeping cables organized and out of the plant's light path.

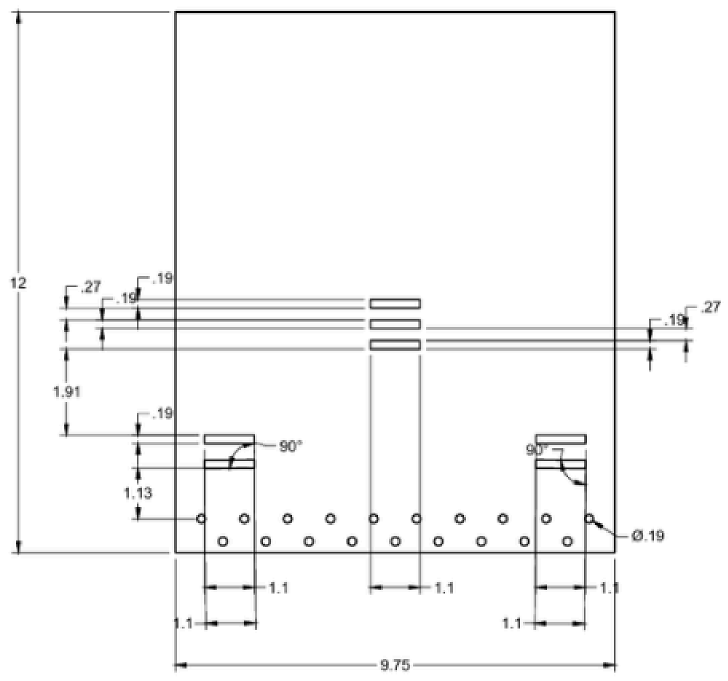
Component	Quantity	Dimension
Lid Cover Main	1	 10 8.8 6.56 5.46 4.91 4.77 0 Ø.06 0 1.71 4.16 4.6 5.4 5.84 8.29 10

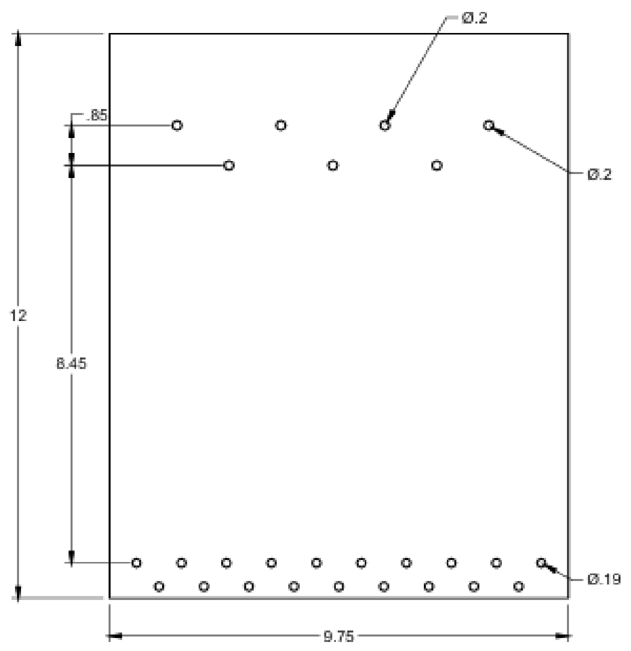
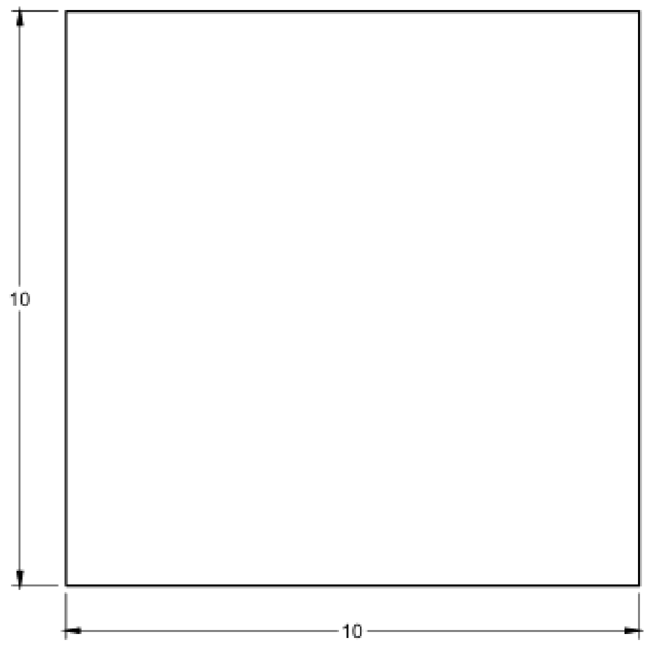
Lid Cover Wire Slot	1	
Lid Cover Top	1	
Lid Cover Bottom	1	

Plant Enclosure

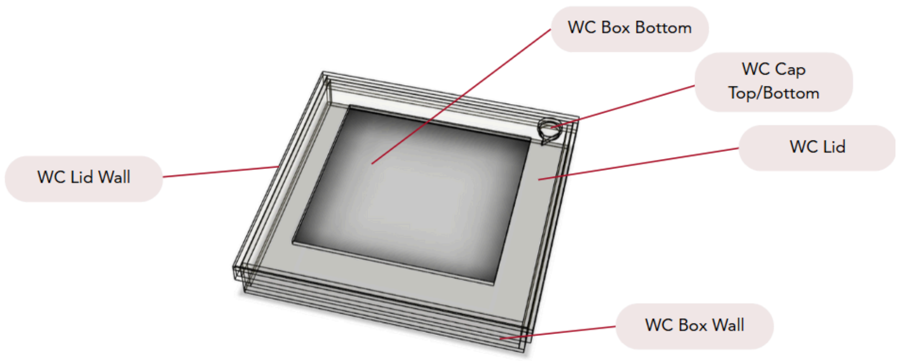


The enclosure body is composed of five structural pieces: 3 Humidity/Temperature Sensor Walls, 1 Soil Moisture Sensor Wall, and 1 Enclosure Bottom. The three pieces assemble to form a 12 in × 10 in × 10 in acrylic box that houses the plant and support sensor integration. In addition, all the enclosure walls feature small holes along the bottom edges. These holes enable capillary water transfer from the water compartment below, allowing moisture to rise into the soil without oversaturating the top surface or damaging the sensors.

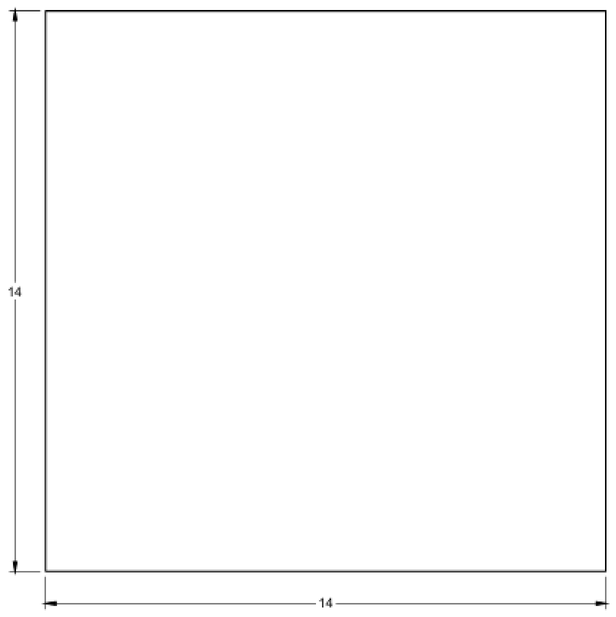
Component	Quantity	Dimension
Soil Moisture Sensor Wall	3	 12 .27 .19 .19 1.91 .19 90° 1.13 90° .19 .27 Ø.19 1.1 1.1 1.1 1.1 1.1 9.75

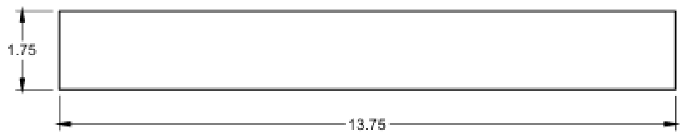
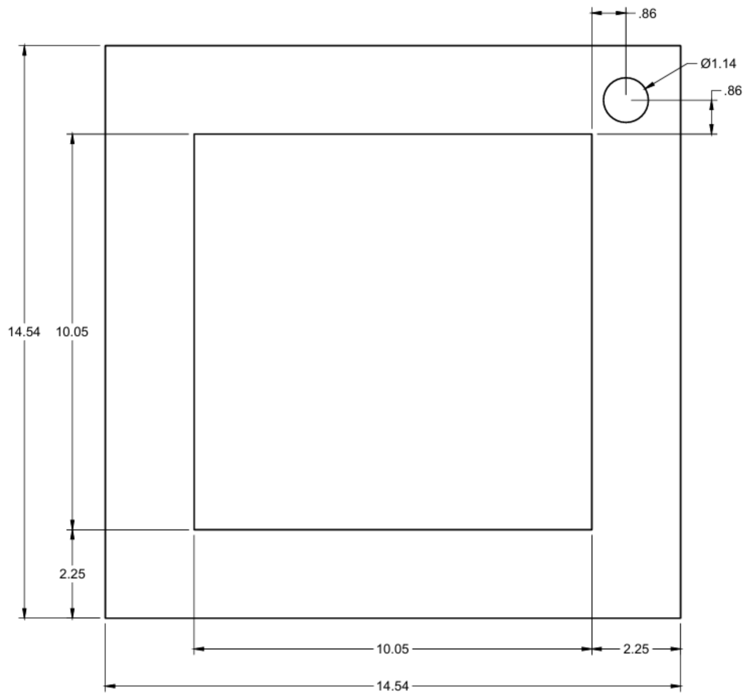
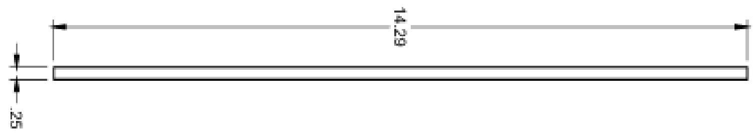
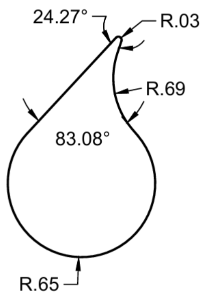
Humidity/Temperature Sensor Wall	1	
Enclosure Bottom	1	

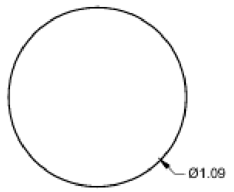
Water Compartment (WC)



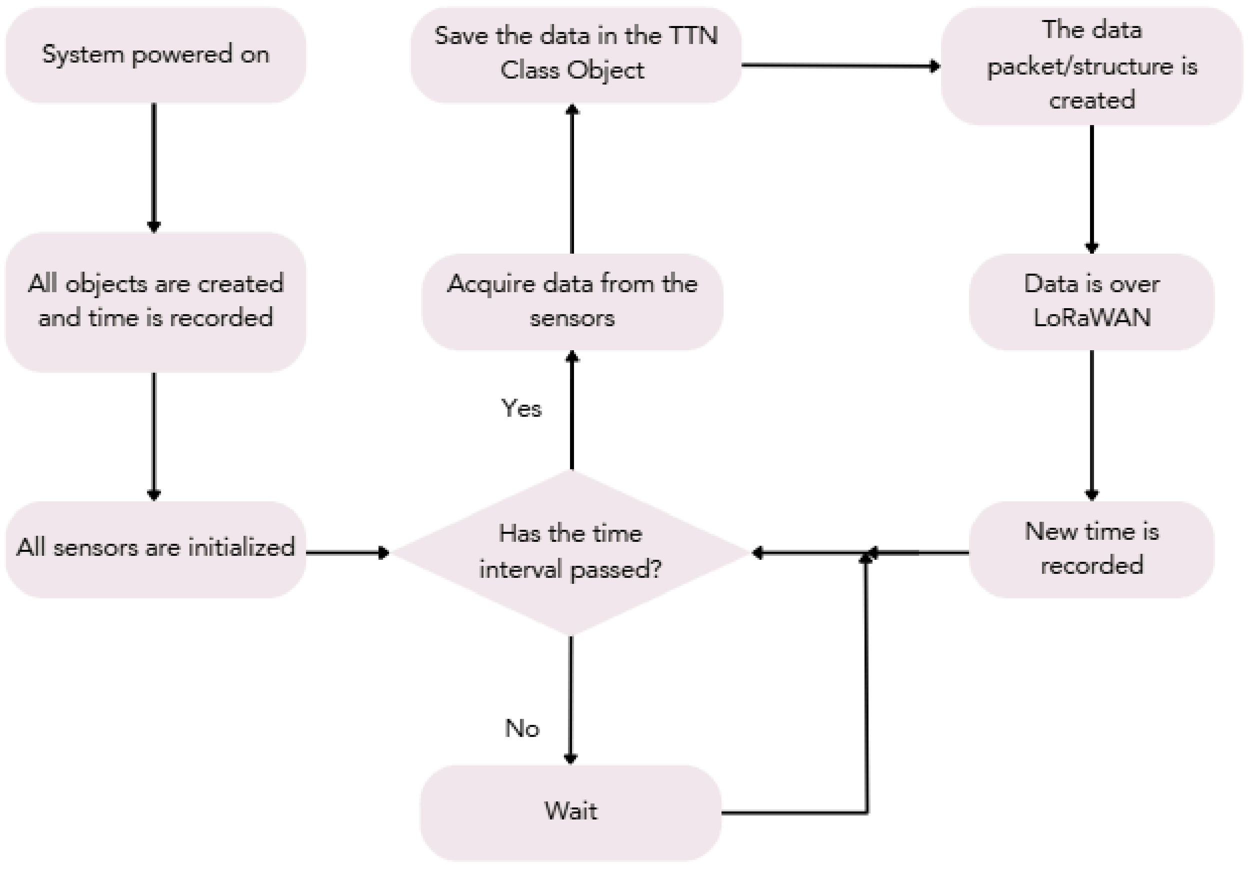
The Water Compartment (WC) is constructed by attaching the four WC Box Walls to the WC Box Bottom, forming the main structural water container. Resting on top of this assembly is the WC Lid, which is framed by its own set of four WC Lid Walls. The WC Lid features a dedicated water inlet, allowing students to pour water into the compartment safely without needing to remove the lid. This inlet is sealed with a custom cap, composed of two parts: a bottom circular base that fits securely into the inlet opening and a top piece shaped like a water droplet, visually indicating the purpose of the opening and guiding students to the correct location for refilling.

Component	Quantity	Dimension
WC Box Bottom	1	

WC Box Wall	4	
WC Lid	1	 .86 Ø1.14 .86 14.54 10.05 2.25 10.05 2.25 14.54
WC Lid Wall	4	***This image shows the top profile of the component, the component is 1.746 in wide 
WC Cap Top	1	

WC Cap Bottom	1	

Appendix IX: System Block Diagram


System powered on
All objects are created and time is recorded
All sensors are initialized
Has the time interval passed?
Yes
No
Wait
Acquire data from the sensors
Save the data in the TTN Class Object
The data packet/structure is created
Data is over LoRaWAN
New time is recorded

Appendix X: Bill of Materials

Part	Part Number	Item Description	Quantity	Unit Price	Total Price
Amazon	B07KRKVSML	Weld-On #4 Adhesive (4 oz.)	1	16.44	16.44
Amazon	B01786ZU4O	24" x 24" - 1/4" Clear Extruded Acrylic Plexiglass Sheet	***	0.10	113.64
McMaster	1125T412	Marine-Grade Plywood Sheet 12" x 24" x 1/4"	1	18.48	18.48
Wiget Co	6-R000-XXX-CS	Size 000 Cork Stoppers - Standard	25	0.18	4.50
Amazon	B09XH5DRXV	8Pcs Capacitive Soil Moisture Sensor	1	8.99	8.99
DigiKey	1528-2832-ND	Ultrosonic Sensor Distance 3V	1	3.95	3.95
Amazon	B01DKC2GQ0	Digital Temperature Humidity Sensor	1	9.99	9.99
Amazon	B07PCJP9DY	4 pc Solderless Breadboard	1	6.69	6.69
Amazon	B0BRTJXND9	120pcs Multicolored Dupont Wire	1	5.97	5.97
Amazon	B014MJ8J2U	Soil Moisture Reader	1	12.99	12.99
Amazon	B0CXHV22GD	Humidity/Temperature Sensor	1	5.90	5.90
Amazon	B07CJYSL2T	Breadboard Wires	1	9.99	9.99
McMaster	90710A145	18-8 Stainless Steel Thin Profile Hex Nut	1	4.36	4.36
McMaster	91292A311	18-8 Stainless Steel Socket Head Screw	1	10.36	10.36
				Total Cost	232.26

***Price calculated at $0.10 per square inch of the acrylic material

Appendix XI: TTN Login

Login: genevahsplant
Password: Mswalkeriscool25

Keys can be obtained by logging into the TTN application

Appendix XII: Organization and Team Timeline

Organization

Project Scope/Student Involvement

The project involves a group of four students working collaboratively to achieve the outlined goals. Our team is responsible for creating a system that will interact with students and engage them in IoT learning.

Timeline and Coordination

1. Weekly Meetings: A when2meet poll was been sent to all group members to coordinate weekly meetings with group members and teachers. (This ensures consistent communication and progress.) The group also uses the allotted class time to work and discuss the project.
2. Communication with Community Partner: Regular updates and discussions with the community partner, Kirsten Abbott, will be scheduled to align the projects with community needs and expectations.

Roles and Contacts

1. Student Team Members
    a. Student 1: Brandon Feraud-Solorzano, bf282@cornell.edu
    b. Student 2: Miriam Moonga, mm2564@cornell.edu
    c. Student 3: Aya Mtume, ajm448@cornell.edu
    d. Student 4: Cira Diop, cd623@cornell.edu

2. Community Partner:
    a. Kirsten Abbott, kabbott@genevacsd.org
        i. Will provide guidance, resources, and feedback to ensure the project aligns with community goals.
    b. Shelley Walker, shelley.walker@genavacsd.org
        i. A botany teacher whom we were connected to through Mrs. Kirsten Abbot,t and will allow us to actively participate with her students through this project

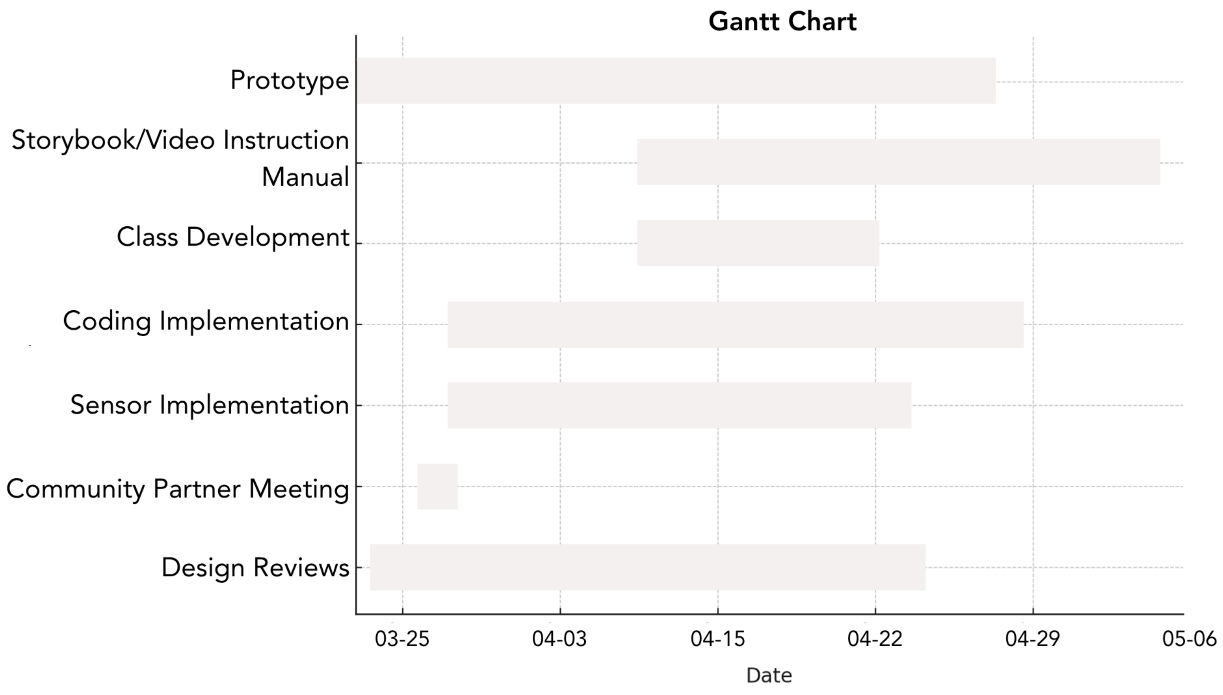
3. Point of Contact (POC)
    a. Cira Diop
        i. Coordinated communication and meetings with the community partner, Kirsten Abbott, as well as the IoT partner team, and the STEM Teacher.
        ii. Designed and developed the CAD model for the Plant Monitoring Enclosure.
        iii. Prepared components for assembly by laser-cutting the enclosure pieces.
        iv. Managed part orders with teacher assistant, Mike Liao.
    b. Brandon Feraud-Solorzano
        i. Assisted with hardware integration, electrical testing, and sensor calibration.
        ii. Contributed to TTN data platform integration and system documentation.
    c. Miriam Moonga:

    i. Assisted with hardware integration, electrical testing, and sensor calibration.
    ii. Contributed to TTN data platform integration and system documentation.
    iii. Responsible for developing the website for visualizing real-time plant monitoring data.

d. Aya Mtume
    i. Researched Social Context
    ii. Created the Instructional Manual
    iii. Coordinated with Ms. Walker for scheduling meetings, and communicated with Ms. Walker to get progress in the Plant Monitoring System in the classroom

All remaining aspects of the project, including design decisions, assembly, testing, and final presentations, were completed collaboratively by the team.

Appendix XIII: Timeline



Task Name	Start Date	End Date	Duration	Progress (%)
Design Reviews	2025-03-21	2025-04-25	3 days	100
March Design Review	2025-03-21	2025-03-21	1 day	100
April Design Review	2025-04-25	2025-04-25	1 day	100
Final Design Review	2025-05-11	2025-05-11	1 day	100
Community Partner Meeting	2025-03-24	2025-03-26	2 days	100
Meeting 1	2025-03-26	2025-03-26	0 days	100
Meeting 2	2025-03-24	2025-03-24	0 days	100
Sensor Implementation	2025-03-26	2025-04-24	21 days	100
Research and Discuss Sensors	2025-03-26	2025-03-28	3 days	100
Order Sensors	2025-04-07	2025-04-07	0 days	100
Wire Sensors to Featherboard	2025-04-18	2025-04-21	2 days	100
Calibrate Sensors	2025-04-22	2025-04-23	2 days	100
Sensors Implementation Complete	2025-04-24	2025-04-24	0 days	100
Coding Implementation	2025-03-26	2025-05-01	26 days	100
Discuss Code and Define Architecture	2025-03-26	2025-03-28	3 days	100
Class Development	2025-04-07	2025-04-22	12 days	100
Sensor Class	2025-04-07	2025-04-16	8 days	100

TTN Class	2025-04-15	2025-04-22	6 days	100
Main Coding Logic (Student Facing)	2025-04-18	2025-04-25	6 days	100
Data Analysis Coding Logic (Teacher Facing)	2025-04-25	2025-04-30	4 days	100
Class Development Complete	2025-05-01	2025-05-01	0 days	100
Storybook/Video Instruction Manual	2025-04-07	2025-05-09	24 days	100
Create a Storyboard + Plan	2025-04-07	2025-04-09	3 days	100
Compose an instruction manual	2025-04-10	2025-04-25	12 days	100
Print, Inspect, and Rehearse Curriculum	2025-04-28	2025-04-30	3 days	100
Field Testing	2025-05-09	2025-05-09	0 days	100
Prototype	2025-03-20	2025-04-29	29 days	100
CAD Design	2025-03-20	2025-03-28	7 days	100
Order Parts	2025-04-07	2025-04-17	9 days	100
Coordinate with the Story Board	2025-04-07	2025-04-09	3 days	100
Print and Assemble	2025-04-17	2025-04-23	5 days	100
Structural test	2025-04-28	2025-04-29	2 days	100
Assemble with sensors	2025-04-28	2025-04-29	2 days	100
Field Testing @ Geneva High School	2025-05-09	2025-05-09	1 day	100

Appendix XIII: Instruction Manual

Introduction to the Internet of Things (IoT)

IoT is a system where devices can collect and exchange data over the internet. You might be using IoT in your daily life without even realizing it! From smart thermostats to Apple Watches to self-watering gardens, IoT is transforming how we interact with our environment. In this project, you'll learn how to build your own IoT system. This manual is created as a guide for an IoT application of your choosing. For this project, we are creating an IoT system to monitor and analyze plant growth.

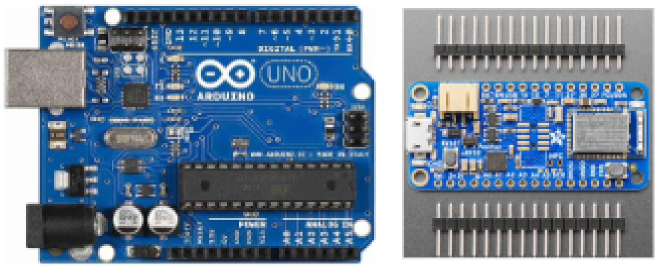
Physical Built | Not all IoT systems require a physical structure, but this project includes an enclosure, hardware, and software integration. The enclosure is pre-built, with designated slots for sensors. Soil moisture sensors can be inserted into the side slots, the distance sensor must be screwed onto the lid, and the humidity/temperature sensors can be placed through the holes on the three body walls of the enclosure.

Collecting Data | You'll use sensors to collect data about soil moisture, environmental temperature and humidity, and distance (plant growth). Then, you'll send this data to the cloud using a microcontroller, so you can monitor it from anywhere!

What Is Arduino? | Arduino is like a small computer that you can program things like lights, sensors, and motors. It has two main parts:

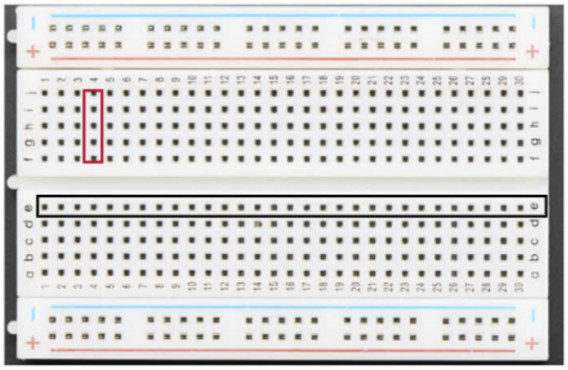
- Hardware: The physical board that you connect your sensors and devices to. It includes a microcontroller that runs your code and pins for wiring the sensors.
- Software (IDE): This is the Arduino IDE (Integrated Development Environment), where you write code and upload code to the board.

Compatible boards | Instead of a standard Arduino, we're using the Adafruit Feather Board. It's smaller, lighter, and uses less power. Depending on your needs, you can choose boards of various sizes, USB types, and power levels.



Arduino Board and Adafruit Feather Board

Breadboards | A **breadboard, also known as the solderless board,** is a tool used to build prototypes of electronic circuits without having to solder. It helps you connect different electronic components like sensors, LEDs, and microcontrollers.



In this image the horizontal rows (red) are numbered.
The vertical columns (black) are labeled with letters.

Key Concepts:

- **Horizontal holes** (rows) are connected electrically, but not across the middle gap (for example: pins f4 to j4 are not connected to pins a5 to e4)
- **Vertical holes** (columns) are not connected
- **Buses (Power Rails)**: Run along the sides and are used to distribute power (usually marked with red for positive and blue for negative).
- Ohm's Law: Voltage = Current x Resistance
  - Voltage: The force that transmits the current through the circuit.
  - Current: How much flow of electric charge flows?
  - Resistance: How much the current is being slowed down or resisted.
- **Connecting in Parallel**: All components share the same voltage, but the current splits. Great for sensors!
  - Connecting resistors in parallel allows for multiple paths for current to flow through a circuit
  - The total resistance is calculated by adding the reciprocals of the resistances (i.e., 1/R1 + 1/R2 + 1/R3 = Rtotal)
- **Connecting in Series**: Components are connected one after another. Not used often in sensor circuits.
  - Connecting resistors in series allows for the same amount of current to flow through each one
  - This increases the total resistance of the circuit
  - The total resistance is calculated by adding the resistances of each resistor

Sensors | There are many types of sensors. Depending on your application, you will need a specific sensor. Sensors work like our ears, we listen, and we send to our brain to interpret. Here is information on the sensors used on the Plant Monitoring System

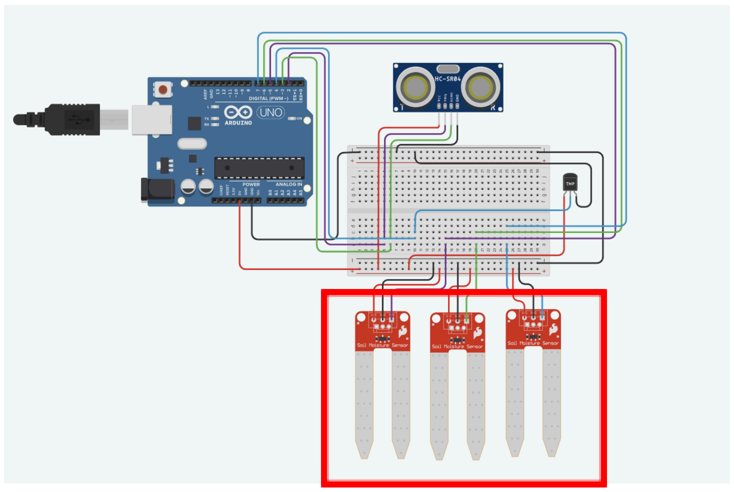
Sensor #1: Soil Moisture Sensor | Capacitive Soil Moisture Sensor

This sensor measures how much moisture is in the soil, essentially telling you when your plant needs water. It uses dielectric theory in order to give an analog value associated with the moisture percentage of the soil. Thankfully, we don't need to know about all of that! Just know that this sensor will allow us to have some value associated with the moisture level of the soil in the enclosure.

Based on our Circuit Diagram, we can see how we wire these 3-pin connections:

- VCC
- GND
- AOUT

As shown on the following circuit diagram, the pin connections correspond to the placements on the featherboard and breadboard. There are three soil moisture sensors shown in this diagram, each with three pin connections to the 5V terminal, ground connection, and pins 5,6, and 7. VCC connects to the power supply (5V), GND connects to the ground terminal, and AOUT connects to the corresponding 5,6, and 7 pins.



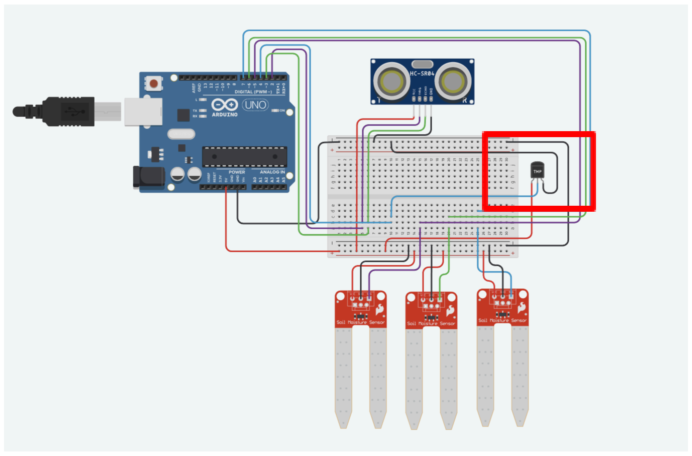
How to Connect |As you can see, the pin connections aren't directly attached to the feather, and instead are connected in series by wires. We can do this as those holes are connected electrically and therefore don't need to be directly connected.

Sensor #2: Temperature & Humidity Sensor | DHT11 Sensor Module

This sensor lets you monitor the environment around your plant. We can use this sensor to get information about the environment the plant is in and determine what are ideal conditions are and how to manipulate the environment to achieve that!

Connections (3 Pins):

- VCC
- GND
- DATA



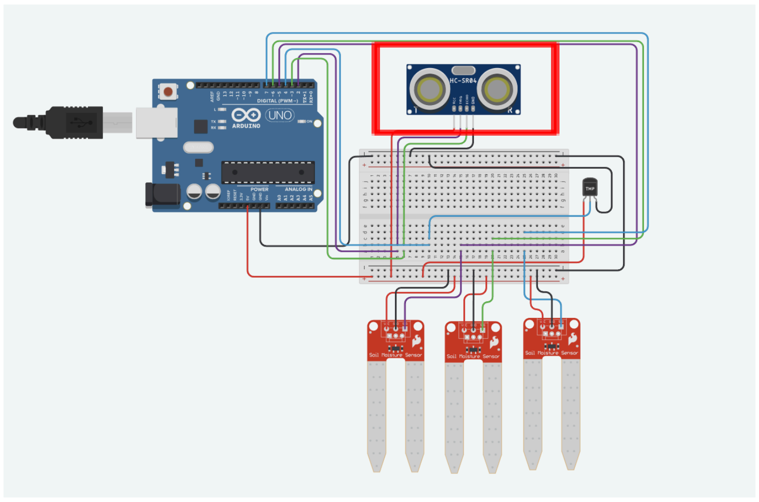
How to Connect | This connection is very similar to the soil moisture sensor connection, as there are 3 pins corresponding to ground, power supply, and a pin. The pin used in this diagram is pin 4, with the ground and VCC connections directly connecting to the power buses.

Sensor #3: Distance Sensor | Adafruit Distance Sensor - TSL2591

This sensor detects the distance between the sensor itself and the top of your plant, in this case, measuring the longitudinal growth of your enclosed plant. Hopefully, the plant will grow straight and allow for the distance sensor to detect the top of the plant as a measure of its growth over time.

Connections (4 Pins):

- VCC
- GND
- TRIG
- ECHO



How to Connect | This sensor is a little different, as it has 4 pins as opposed to 3. This is because there is a trigger pin, as well as an echo pin. This sensor works by sending an ultrasonic sound pulse, traveling through the air like a "ping." Once the sound hits an object, the sound bounces back and is detected through the echo. The distance is then calculated by using the time it took for the ping to travel back to the sensor, which is why we need a connection for both the trigger or "ping" and the echo, utilizing pins 2 and 3, respectively.

Arduino Code + TTN (The Things Network) Integration

Once the sensors are wired up, it's time to write and upload your Arduino code. We have provided code for you to monitor plant growth and send data over TTN!

What is TTN? | The Things Network (TTN) is a global network that allows wireless devices to send data to the internet using LoRaWAN (Long Range Wide Area Network), connecting the devices without Wifi. After we collect our data, it is sent over LoRaWAN where a TTN gateway can pick up the signal. This gateway receives the data then connects to the internet, where it sends the data to TTN. Then with our TTN account we can decode the data and create graphs using the data we collected ourselves!

Explanation of Code | A class in C++ is like a blueprint for creating objects. Think of it like designing a car: the class describes what the car has (tires, engine) and what it can do (drive, honk). In Arduino, you create an object from a class and call its functions. Our project uses parent and child classes, where the parent class contains global attributes and functions like initializing the sensor, and each sensor is represented by a child class that has functions specific to our project!

Here we can see our parent and child classes, as well as their attributes!

Parent Class: Sensor

Child Classes:

- distanceSensor(trigger Pin, echo Pin)
- tempHumiditySensor (analog Pin)
- soilMoistureSensor (analog Pin)

Class Attributes

Global Attributes:

- begin()
- sensorType()

Child class specific attributes:

- distanceSensor:
  - readData()

- tempHumiditySensor:
  - readTempData()
  - readHumidityData()

- soilMoistureSensor:
  - readData()

TTN Class | In order to package the data to be sent over TTN, we created a TTN class to easily send the data. The class is as follows:

Class: ttnData

Class Attributes: floats (temp, humidity, distance, moisture)

- Part of the data packet we send over LoRaWAN

Class method: sendPacket()

- Sends the data packet over LoRaWAN

Now We Start the Project | With your understanding of the circuit diagram, sensors, and classes, we can put all of this together to get the project running!