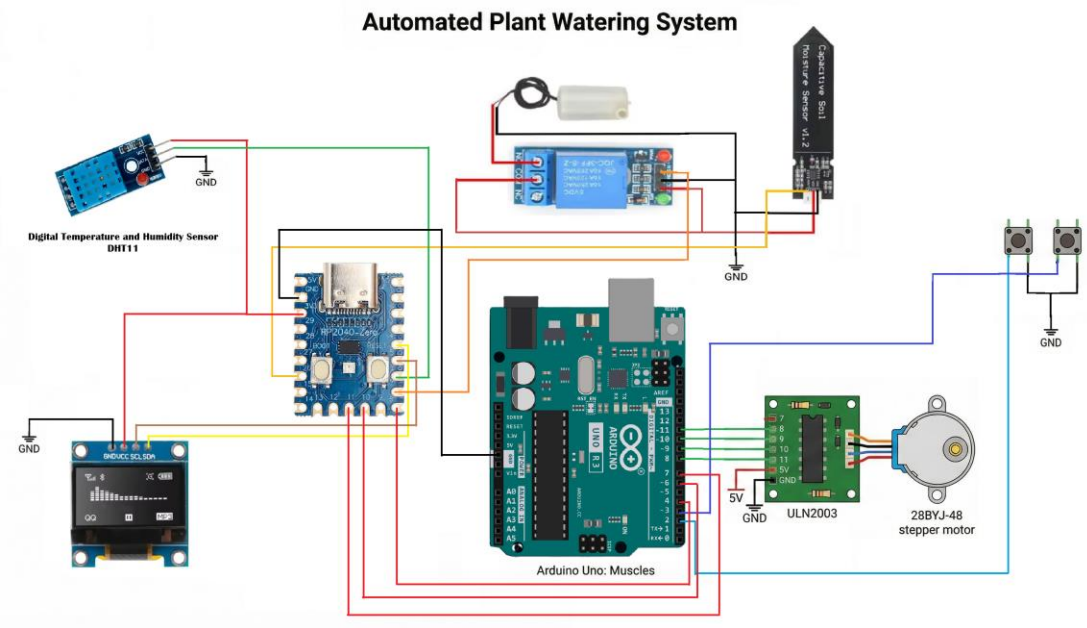
Automatic Plant Watering Elephant with Rotating Table

ADA525 Final Portfolio Report

Author: Carlos Carpio **Date:** December 2025 **Institution:** HVL - Western Norway University of Applied Sciences **Course:** ADA525 - Advanced Data Acquisition

---

Introduction



Visualization of the Automatic Watering Elephant project, showing the integration of nature and technology in a home plant care system.

Q0. Project Description

ELEPHANT AND DETACHABLE HAT CONCEPT

Figure 1: Concept art for the "Watering Elephant" design, featuring a decorative elephant figurine with a detachable hat that houses the electronics. The playful design makes the functional watering system blend naturally into a home environment. Generated using Google Gemini AI, nanoBanana.

The **Automatic Watering Elephant** is a dual-microcontroller plant care system that autonomously waters two avocado plants from Peruvian seeds using a rotating table mechanism. The system is designed to solve a real problem: maintaining healthy plants while traveling or during busy periods.

Why I wanted to make it: Growing avocado plants from seed requires consistent moisture levels. After losing several promising seedlings to irregular watering during travel, I was motivated to create a system that could reliably care for my plants without daily intervention. The project also presented an exciting opportunity to integrate digital fabrication, embedded systems, and inter-device communication.

Key Features:

1. **Dual-Controller Architecture:** RP2350-Zero handles sensing and decision-making ("Brain"), Arduino Uno controls mechanical rotation ("Muscles")
2. **Single Sensor for Multiple Plants:** A rotating table brings each plant to the watering position, optimizing hardware usage
3. **Capacitive Soil Moisture Sensing:** Monitors soil moisture with 12-bit ADC resolution
4. **Environmental Monitoring:** DHT11 sensor tracks temperature and humidity
5. **Pendulum Motion:** The table swings back and forth (not continuous rotation) to prevent cable tangling
6. **Flexible Position Learning:** User can set plant positions in any order using teach-mode buttons
7. **OLED Display:** Real-time status visualization

Project Evolution Timeline

This project evolved through multiple phases during the ADA525 course:

Phase	Date	Focus	Key Learning
Initial Concept	Sep-Oct 2025	Single-plant Arduino prototype	Basic pump control, sensor reading
Intermediate Report	Nov 2025	RP2350-only design with rotating table	Power management challenges, motor interference
Final System	Dec 2025	Dual-controller architecture	Inter-MCU communication, voltage level shifting

Evolution Highlights:

- Started with a simple Arduino-only watering system for one plant
- Teacher feedback pushed for more complexity → added mechanical rotation
- Power instability in RP2350-only design led to dual-controller architecture
- Cable tangling issue led to pendulum motion innovation
- Component failures taught importance of quality parts and debugging methodology

The final system represents the culmination of lessons learned from each iteration, where early failures directly informed the robust design choices in the production version.

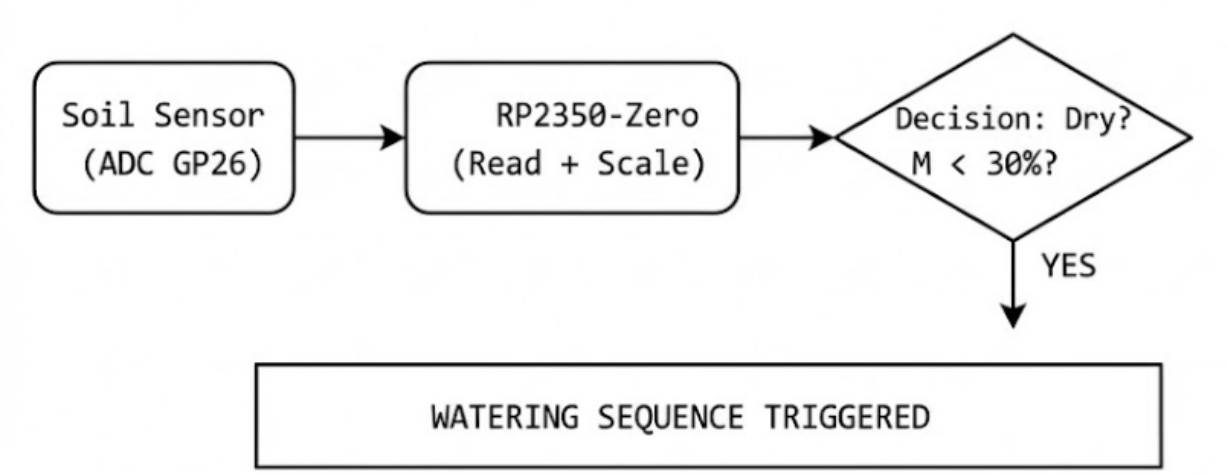
Q1. Core Workflow

The system workflow operates as follows:

User Interaction:

1. User powers on the system
2. LED illuminates indicating Learning Mode
3. User HOLDS Button A/B to manually rotate table to first plant position
4. User CLICKS to save position (LED blinks 3×)
5. User moves to second plant and CLICKS to save (LED blinks 5×)
6. System enters autonomous Run Mode

Sensing and Processing:



System Response (Handshake Protocol - Modified):

1. RP2350 sets `WATER_SIGNAL = HIGH` → Arduino receives on Pin 4 (Trigger)
2. Arduino rotates table to Plant 1
3. **Arduino activates pump (Pin 12)** for **3 seconds**
4. Arduino rotates to Plant 2
5. **Arduino activates pump (Pin 12)** for **3 seconds**
6. System enters sleep mode (waiting for next trigger or reset)

Figure 2: Screenshot of the Arduino Serial Monitor during a complete watering cycle. The output demonstrates the flexible position learning system. The users can set positions in any order (A→B or B→A) depending on the motor's starting position. After both positions are saved (with LED blink feedback), the simulation sequence automatically begins: the table moves to Position A, waters for 3 seconds, moves to Position B, waters again, and enters sleep mode.

NOTE: Pump control was moved from RP2350 to Arduino (Pin 12) to overcome GPIO latching issues on the prototype board. The RP2350 retains the role of "Brain" (Sensing), while Arduino acts as "Muscle" (Motion & Action).

System Implementation

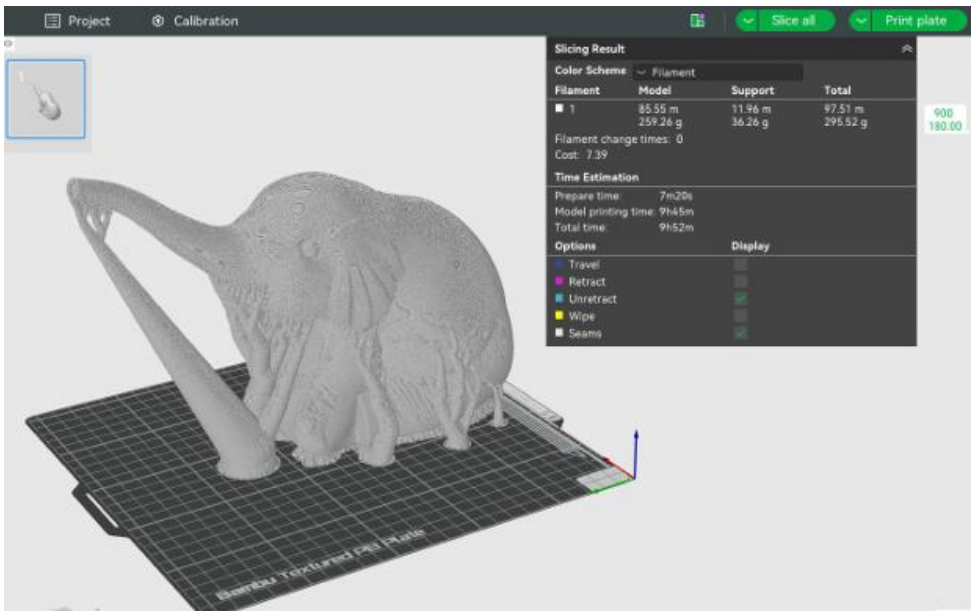
Digital Fabrication Focus

Q2. CAD & 3D Printing Integration

The Elephant Figurine Evolution:

From idea to 3D modeling, I started by generating the elephant concept image in Google Gemini AI and then used MakerWorld's MakerLab "Image to 3D" tool to turn it into a 3D mesh. I first imported this mesh into Fusion 360 to inspect and split the geometry, but the dense mesh was difficult to handle there, so I moved the project to Blender.

In Blender, I used the Boolean modifier to subtract an inner shape from the elephant, creating a clean internal cavity and turning the solid model into a hollow "elephant kettle" that could house the electronics. Blender also helped me fix errors and rebuild the imperfections from the AI-generated mesh. In the end, I spent more time repairing than building, but that taught me valuable lessons as an engineer about real-world challenges and different approaches to problem-solving.

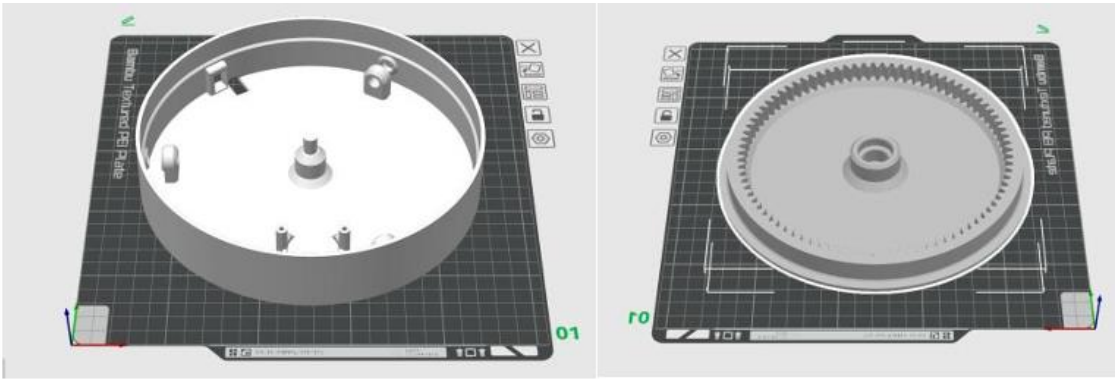


Blender Mesh Repair Process

Figure 3: When I import a model from Blender into Bambu Studio it often shows up much smaller, because the two programs use different unit scales. If I do not fix the size right away, later edits and slicing go wrong, and I kept running into the same error after shrinking the model. Before scaling to real size I first cut the hat away from the elephant in Bambu Studio, so I can scale and align each part separately with better precision.

The Rotating Table Mechanism:

The rotating table mechanism required custom gears designed in CAD and manufactured using 3D printing:



3D Model of Rotating Table

Figure 4: CAD model of the rotating display table designed in Fusion 360. The design features an integrated ring gear mechanism that meshes with a small driver gear attached to the stepper motor, providing smooth and precise rotation.

Design Evolution:

Version	Design	Issue	Solution
V1	Single gear, direct drive	Insufficient torque	Added gear reduction
V2	15:85 tooth ratio	Gear slippage	Redesigned tooth profile
V3 (Final)	Optimized spur gear + ring gear	N/A	Successful

COMPONENTS DESIGNED AND PRINTED:

- **Small Driver Gear:** 15-tooth spur gear (orange PLA), press-fit to motor shaft

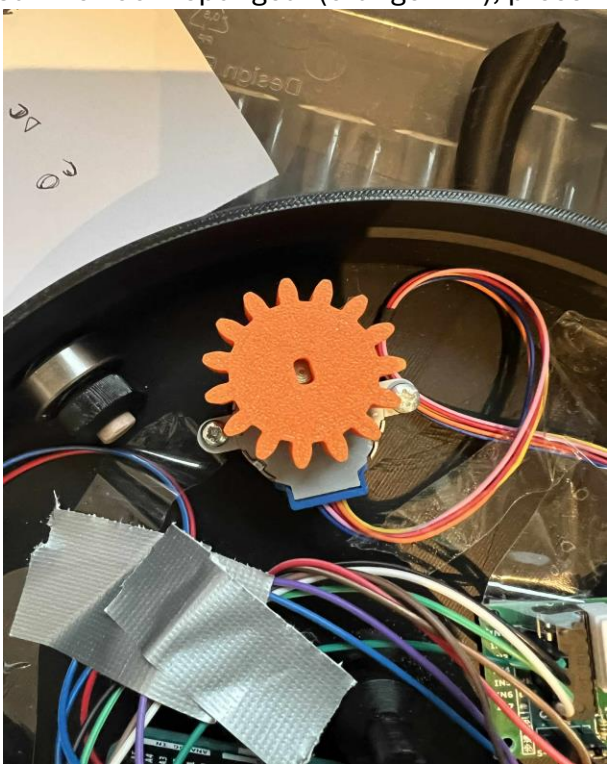


Figure 5: The 15-tooth driver gear printed in orange PLA, designed to press-fit onto the 28BYJ-48 motor's asymmetric shaft.

- **Large Ring Gear:** 85-tooth internal ring gear (white PLA), integrated into table base

Figure 6: The 85-tooth ring gear printed in white PLA, which forms the base of the rotating table and provides the 5.67:1 gear reduction.

Gear Ratio Calculation:

$$\text{Gear Ratio} = \frac{85}{15} = 5.67$$

This provides:

- **Speed reduction:** Motor runs 5.67× faster than table rotation
- **Torque multiplication:** Table receives 5.67× motor torque
- **Precision:** 23,210 steps per table revolution (0.0155° resolution)

Tools Used:

- Fusion 360 for gear modeling
- Cura slicer for print preparation
- Creality Ender 3 printer (0.2mm layer height, 20% infill)

Q3. Fabrication Challenges

Challenge 1: Gear Backlash Early prints exhibited significant play between meshing teeth, causing positioning inaccuracy.

Solution: Adjusted tooth clearance in CAD by reducing the module parameter by 0.1mm. Required 3 print iterations to achieve acceptable fit.

Challenge 2: Motor Shaft Coupling The 28BYJ-48 motor has an asymmetric shaft that was difficult to model accurately.

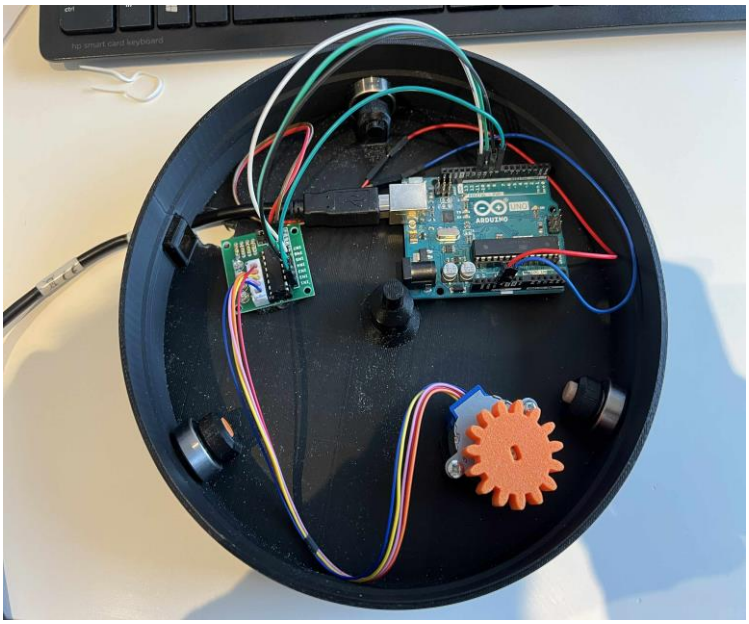
Solution: Created a test coupler with incrementally different hole shapes, then updated the main gear design based on best-fit results.

Challenge 3: Ring Gear Warping The large ring gear (120mm diameter) exhibited warping during printing due to uneven cooling.

Solution: Enabled heated bed adhesion rim and printed at reduced speed (30mm/s) for first 5 layers.

Physical Computing Focus

Q4. Hardware Integration



Internal Electronics Layout

Figure 7: Internal view of the electronics assembly showing the RP2350-Zero microcontroller, relay module, and sensor connections. The compact layout fits within the elephant figurine's detachable hat.

The system incorporates multiple physical computing elements across two microcontrollers:

RP2350-Zero (Sensor Hub & Brain):

Component	Purpose	Connection	Justification
Capacitive Soil Sensor	Moisture detection	GP26 (ADC)	Corrosion-resistant, long-term reliability
DHT11	Temp/Humidity	GP6 (Digital)	Low cost, sufficient accuracy for plant care
SSD1306 OLED	Status display	I2C (GP4/5)	Low power, always-on visibility
Relay Module	Pump control	GP7	Isolates high-current load from MCU

Arduino Uno (Motor Controller + Pump):

Component	Purpose	Connection	Justification
28BYJ-48 Stepper	Table rotation	Pins 8-11	High torque, low cost, easy control
ULN2003 Driver	Current amplification	Pins 8-11	Matches motor requirements exactly
Push Buttons ×3	Position A, B, Reset	Pins 2, 3, 5	Simple, reliable user input
Relay Module	**Pump control**	**Pin 12**	**Arduino-direct control for reliability**

Final Implementation: During integration testing, RP2350 GPIO issues led to moving pump control directly to Arduino Pin 12. This simplified the system and improved reliability while maintaining all functionality.

OLED Display

Figure 8: The SSD1306 OLED display (128×64 pixels) showing real-time system status. The display provides immediate visual feedback including current state (Idle/Watering), temperature, humidity, and soil moisture readings. Connected via I2C to the RP2350-Zero (GP4/GP5), this low-power display remains always-on for easy monitoring.

Key Code Snippet - Stepper Control:

```
// Half-step sequence for smoother motion (8 phases)
const int halfStepSequence[8][4] = {
  {1,0,0,0}, {1,1,0,0}, {0,1,0,0}, {0,1,1,0},
  {0,0,1,0}, {0,0,1,1}, {0,0,0,1}, {1,0,0,1}
};

void stepMotor(bool clockwise) {
  stepPhase += clockwise ? 1 : -1;
  if (stepPhase >= 8) stepPhase = 0;
  if (stepPhase < 0) stepPhase = 7;

  digitalWrite(IN1, halfStepSequence[stepPhase][0]);
```

```
  // ... IN2, IN3, IN4
  delay(stepDelay);  // 1ms for maximum speed
}
```

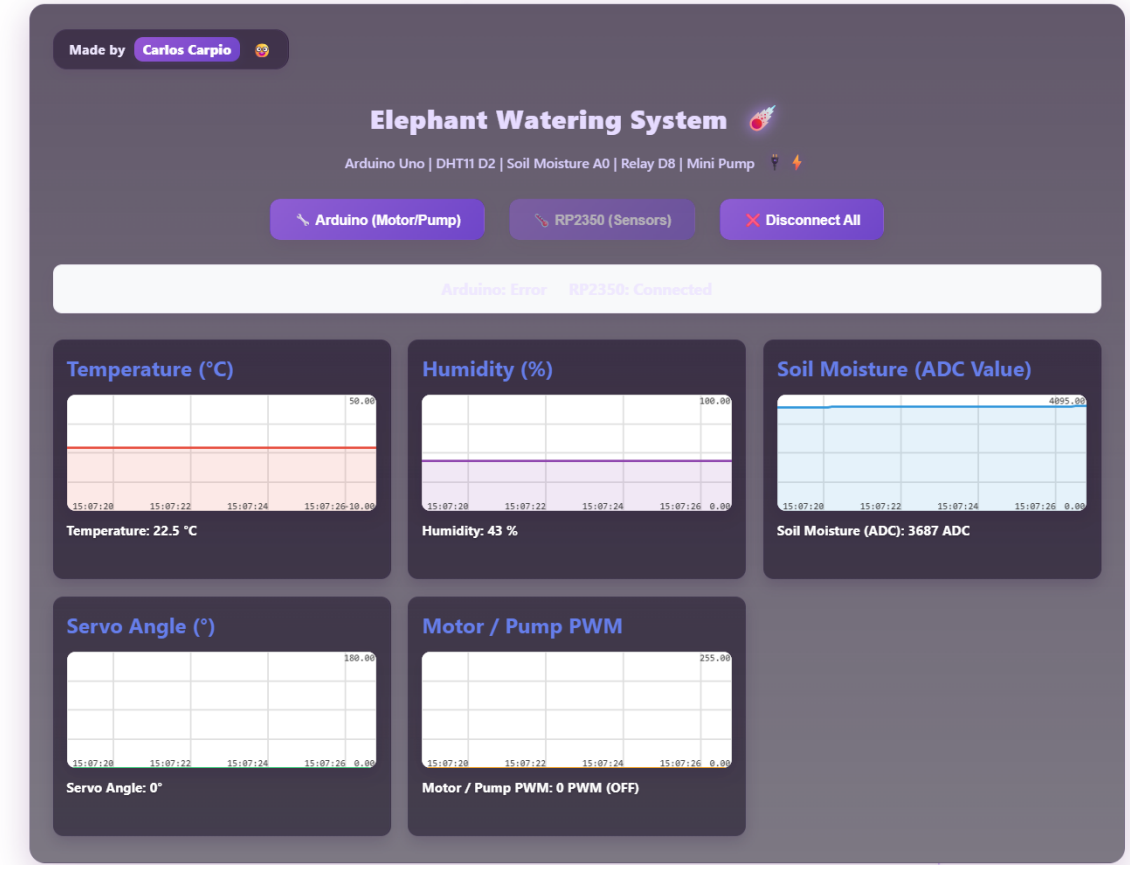
Q5a. Data Acquisition Strategy

Soil Moisture Calibration: The capacitive sensor outputs analog voltage inversely proportional to moisture. Calibration was performed by:

1. Recording ADC value in air: `R_dry ≈ 3000`
2. Recording ADC value in water: `R_wet ≈ 1200`
3. Applying linear interpolation:

$$M = 100 \times \left(1 - \frac{R - R_{wet}}{R_{dry} - R_{wet}}\right)$$

Setting the watering threshold at 30% moisture proved optimal after testing. Lower thresholds (20%) caused stress-visible wilting; higher thresholds (40%) led to overwatering and root rot risk. Avocado plants (*Persea americana*) prefer to dry out between waterings, so a **4-5 day cooldown period** between watering cycles was implemented to prevent overwatering.



Sensor Data Graph

Figure 9: Real-time visualization of sensor data using the "Read Serial Plot" tool created by Professor Frikk Fossdal. This graph demonstrates how the soil moisture ADC values can be monitored live during development and calibration. The tool streams serial output directly into a graphical interface, making it easy to identify sensor behavior patterns and set appropriate thresholds.

Position Tracking: The Arduino maintains a step counter that wraps at 23,210 (one full revolution). When saving positions, the current step value is stored, enabling precise return navigation using the shortest-path algorithm.

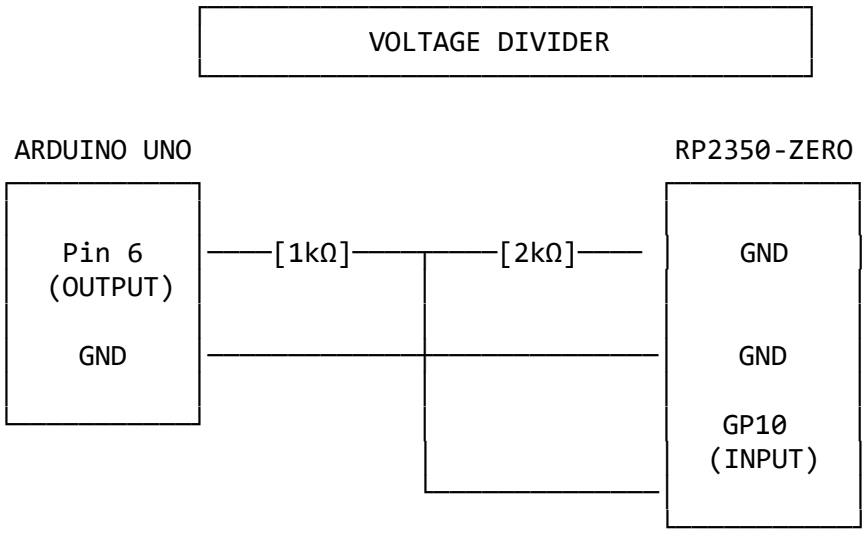
Q5b. Integration Challenges

Challenge 1: Voltage Level Mismatch The RP2350 operates at 3.3V logic; Arduino Uno at 5V. Direct connection from Arduino Pin 6 to RP2350 GP10 risked damaging the Pico.

Solution: Implemented voltage divider (1kΩ + 2kΩ):

$$V_{out} = 5V \times \frac{2000}{3000} = 3.33V$$

Voltage Divider Wiring Diagram:



CAUTION. Never connect Arduino 5V outputs directly to RP2350 GPIO pins! The RP2350 GPIO is rated for 3.3V maximum. Connecting 5V directly will damage or destroy the pin. Always use a voltage divider for signals going FROM Arduino TO RP2350.

Challenge 2: Button Debouncing Initial implementation registered multiple button presses due to mechanical bounce.

Solution: Added 200ms debounce delay and separate timers for position buttons vs. reset button.

Challenge 3: Motor Speed vs. Torque Trade-off Too fast (1ms delay): Motor skipped steps under load. Too slow (5ms delay): Watering cycle took too long.

Solution: Settled on 1ms delay after testing showed the gear reduction provided sufficient torque even at maximum speed.

Challenge 4: Power Supply Voltage Drop (Critical) When both motors (water pump and stepper motor) were connected to the same power supply, the stepper motor stopped working intermittently. Even after adding a 2200µF decoupling capacitor and testing with an L293D driver, the problem persisted.

Root Cause Analysis:

- The water pump draws a large inrush current when starting
- This caused the shared power rail voltage to drop significantly
- The L293D driver adds an additional 1.2-2V voltage drop
- Combined effect: stepper driver received insufficient voltage to function

Solution: Separated the system into **two independent 9V battery supplies**:

Battery	Powers	Reason
9V Battery #1	Arduino Uno + Stepper Motor + ULN2003	Isolated motor power prevents interference
9V Battery #2	RP2350 + Pump + Sensors (OLED, DHT11)	Pump surges don't affect logic circuits

Critical: Both batteries share a common ground to maintain signal integrity for the handshake protocol.

This dual-battery configuration completely eliminated the voltage instability issues.

Challenge 5: Defective Low-Cost Hardware (The "AliExpress Lesson")

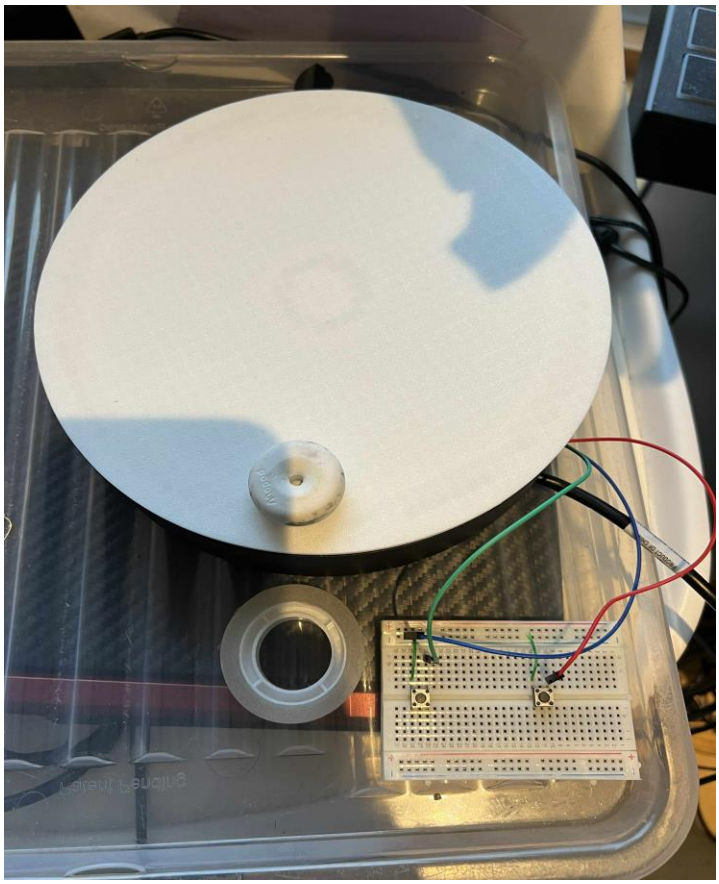
I bought a very cheap RP2350 Zero board from AliExpress to use as the main controller. It behaved unpredictably when driving motors, even with simple code.

- **Symptoms:** GPIO pins got stuck LOW, uploads worked but the program did not run, and motor control was unstable.
- **Tried fixes:** Added big decoupling capacitors, used separate power supplies, and rewrote the code over two very long debugging days.
- **Result:** The board was simply not reliable for heavy work, most likely due to poor internal power design or low manufacturing quality.

Lesson learned: Cheap boards may save money but can cost many hours of debugging. The dual controller setup saved the project, since the reliable Arduino could take over when the "smart" but weak RP2350 Zero failed.

Integration and Synthesis

Q6. Mechanical + Electronic Integration



Final Assembled Rotating Table

Figure 10: The fully assembled rotating table mechanism showing the 3D-printed ring gear, stepper motor mount, and plant positioning area. The pendulum motion design allows back-and-forth movement without cable tangling.

Integrating the 3D-printed gears with the electronic control system required careful consideration:

Physical Constraints:

- Motor mount had to align the gears within 0.5 mm
- Wires had to be routed away from moving parts
- The sensor on the table needed flexible cables

These limits killed my original idea of 360 degree continuous rotation, because the cables kept tangling in early tests. I redesigned the system to use a pendulum style motion instead and rewrote the state machine logic, which made the whole setup more robust.

Design Considerations:

1. **Cable Management:** The pendulum motion (back-and-forth) was specifically chosen to prevent water hose and cable tangling that would occur with continuous rotation
2. **Gear Alignment:** Added adjustment slots in motor mount to fine-tune gear mesh after assembly
3. **Weight Distribution:** Positioned electronics below the table to lower center of gravity

Q7. Iterative Process

Prototype Version 1 - Single Plant (Too Simple):

- Arduino Uno only, no rotating table
- Single plant with direct pump connection
- Manual button trigger for watering
- *Feedback:* Teacher said "too simple" - needed more complexity to demonstrate embedded systems skills
- *Decision:* Add mechanical rotation to water multiple plants with single pump

Prototype Version 2 - Integrated Elephant (RP2350 Only):

- RP2350-Zero as the **sole controller** (no Arduino)
- All components housed inside the elephant figurine's hat
- RP2350 directly driving stepper motor via ULN2003
- Capacitive soil sensor for moisture detection
- Water pump controlled via relay
- *Problem:* Power instability when pump and stepper shared supply - stepper stopped working during pump activation
- *Problem:* Even with 2200µF capacitor and L293D driver, voltage drops persisted
- *Decision:* Split system into two controllers to isolate motor power domains

Prototype Version 3 - Final Dual-Controller System:

- Separated into two controllers: RP2350 (Brain) + Arduino Uno (Muscles)
- Dual 9V battery power supply
- Digital handshake protocol for communication
- Pendulum motion to prevent cable tangling
- Flexible position learning (either button first)
- Full sensor suite: soil moisture, DHT11, OLED display

Major Challenge: Cheap Component Failures

A significant time delay occurred due to defective components purchased from low-cost suppliers:

Failed Component	Symptom	Resolution
Stepper Motor #1	Erratic movement, missed steps	Replaced
Stepper Motor #2	Coil burned, no response	Replaced
DHT11 Sensor	Constant “NaN” readings	Replaced with known-good unit
Dupont Cables	Intermittent connections	Replaced with higher quality cables

Lesson Learned: The time “saved” buying cheap components was lost debugging phantom failures. For future projects, I would invest in quality components from reputable suppliers, especially for critical actuators.

Major Integration Problem:

Issue: When the stepper motor activated, electrical noise caused false readings on the soil moisture ADC, triggering phantom watering cycles.

Diagnosis: Used oscilloscope to observe voltage spikes on the power rail during motor steps. The ULN2003 driver’s inductive kickback was coupling into the shared ground.

Critical Evaluation and Future Scope

Q9. System Evaluation

Strengths:

- Modular dual-controller architecture enables independent testing
- Pendulum motion solves cable management elegantly
- Flexible position learning accommodates various plant arrangements
- Single sensor for multiple plants optimizes cost

Limitations:

- No persistent position storage (lost on power cycle)
- Fixed watering duration regardless of pot size
- Single moisture sensor assumes identical soil for both plants
- No remote monitoring capability

Q10. Future Improvements

Improvement	**Benefit**	**Complexity**
EEPROM position storage	Survives power cycle	Low
Custom PCB Shield	**Eliminates cable mess & improves reliability**	**Medium**
WiFi (ESP32) upgrade	Remote monitoring & control	Medium
Per-plant moisture sensors	Individualized watering	Medium
Water reservoir level sensor	Low-water alerts	Low
Machine learning moisture prediction	Anticipate watering needs	High
Solar power integration	Off-grid operation	Medium

Advancement Priority: With more time, I would first add EEPROM storage for positions (simple code change) and design a **Custom PCB** to replace the breadboard wiring, ensuring a cleaner and more durable system. Then, I would upgrade to ESP32 for WiFi capabilities.

Appendix

A. Pin Configuration Summary

RP2350-Zero:

```
GP26  ← Soil Sensor (ADC)
GP6   ← DHT11 Data
GP4   → OLED SDA (I2C)
GP5   → OLED SCL (I2C)
GP7   → Relay Control
GP8   → WATER_SIGNAL (to Arduino Pin 4)
GP10  ← TABLE_READY (from Arduino Pin 6, via voltage divider)
GP11  → WATER_DONE (to Arduino Pin 7)
```

Arduino Uno:

```
Pin 2  ← Button A (Position)
Pin 3  ← Button B (Position)
Pin 4  ← WATER_SIGNAL (from RP2350 GP8)
Pin 5  ← Button Reset
Pin 6  → TABLE_READY (to RP2350 GP10)
Pin 7  ← WATER_DONE (from RP2350 GP11)
Pin 8  → Motor IN1
Pin 9  → Motor IN2
Pin 10 → Motor IN3
Pin 11 → Motor IN4
```

B. References

1. Arduino Reference. (2025). *Arduino Language Reference*. https://www.arduino.cc/reference/en/
2. Raspberry Pi Foundation. (2024). *RP2350 Datasheet*.
3. 28BYJ-48 Stepper Motor Datasheet
4. ULN2003 Darlington Transistor Array Datasheet
5. DHT11 Humidity & Temperature Sensor Datasheet
6. PlatformIO Documentation. https://docs.platformio.org/