S-Pot: Locomotive Plant Pot

ME 102B Group 29: Katelyn Lee, Matthew Hong, Derek Meadows, Shyan Lee

1. Opportunities

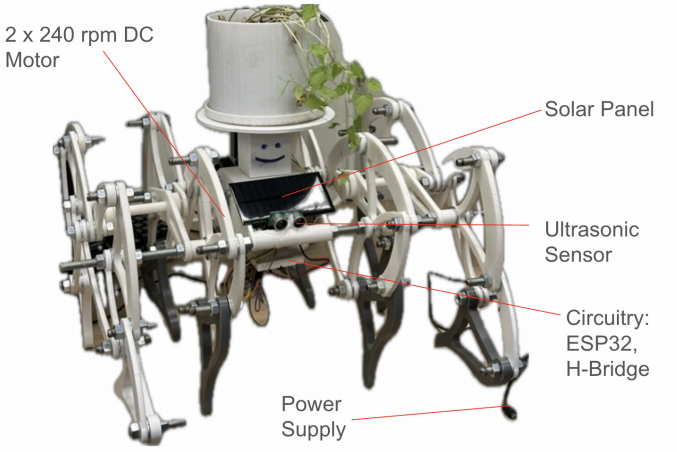
Taking care of plants can be tedious or time consuming for some people. By automating plant care in a playful and aesthetically pleasing way, we hope to offer a charming solution to people who want to avoid the hassle, but enjoy having plants as part of their living space. Our project focuses on creating a locomotive plant pot that can navigate around homes to find sunlight for photosynthesis. Our challenge is for our robot to carry at least 5 pounds, equivalent to a medium-sized house plant.

2. High Level Strategy

The task of creating a locomotive plant care system can be broken down into two parts: (1) **Locomotion** and (2) **Navigation**.

Locomotion is achieved in S-Pot by employing the Theo Jansen walking mechanism. The robot's body contains two DC motors. Each motor actuates a central crank link which imparts rotational motion to the connected links and therefore creates a walking gait that resembles natural leg movement. As the challenge was to carry at least 5 pounds of plant weight, the robot's dimensions had to be relatively large to create a bigger and steadier base for the plant. Given its large size, we chose to use lightweight PLA plastic to create the legs instead of metals to reduce the weight of the robot. The combined heavy weight of the robot and plant also meant that we would need powerful DC motors with higher torque to ensure that the force produced by a motor's crankshaft would be able to overcome the inertia and move the linkages. We settled on a 240 rpm DC motor which showed to be effective in our final device.

Navigation is achieved through a control system that includes an ESP32, a motor driver board, ultrasonic distance sensor and solar panel. The robot is turned on by a power switch, and receives real-time feedback from its environment through the ultrasonic sensor and distance sensor in order to plot its course. The ultrasonic sensor is used for obstacle avoidance wherein the robot is programmed to stop and turn away from the obstacle. The solar panel is used to measure light intensity in the robot's environment. When it receives feedback that the environment is dimly-lit, the robot will move to look for sunlight, and stop in a position where the amount of sunlight is optimal. We have also programmed an upper bound to the optimal sunlight, to ensure that the robot backs away from excessive light, which will dry up the plant. In addition to the solar panel, we wanted to include a moisture sensor to our robot and build a water dispensing station to create an automated watering system. However, due to time constraints, we decided to focus on ensuring the plant receives optimal light, and shelved the watering idea for the time being.

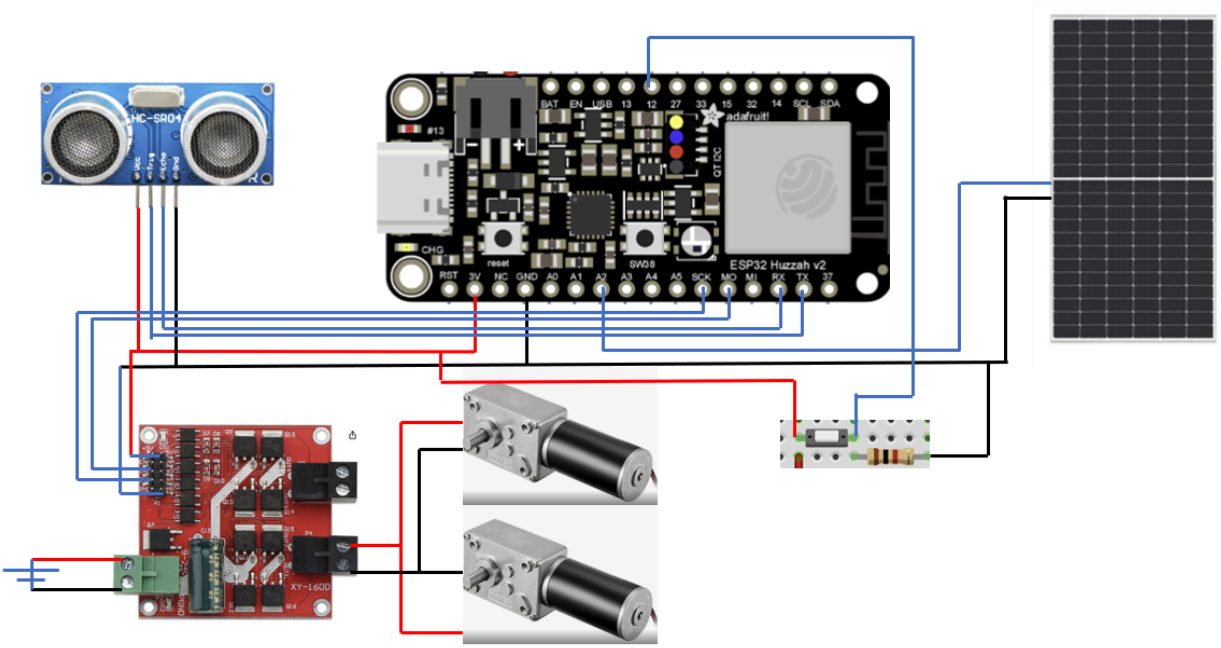


3. Critical Function Calculations

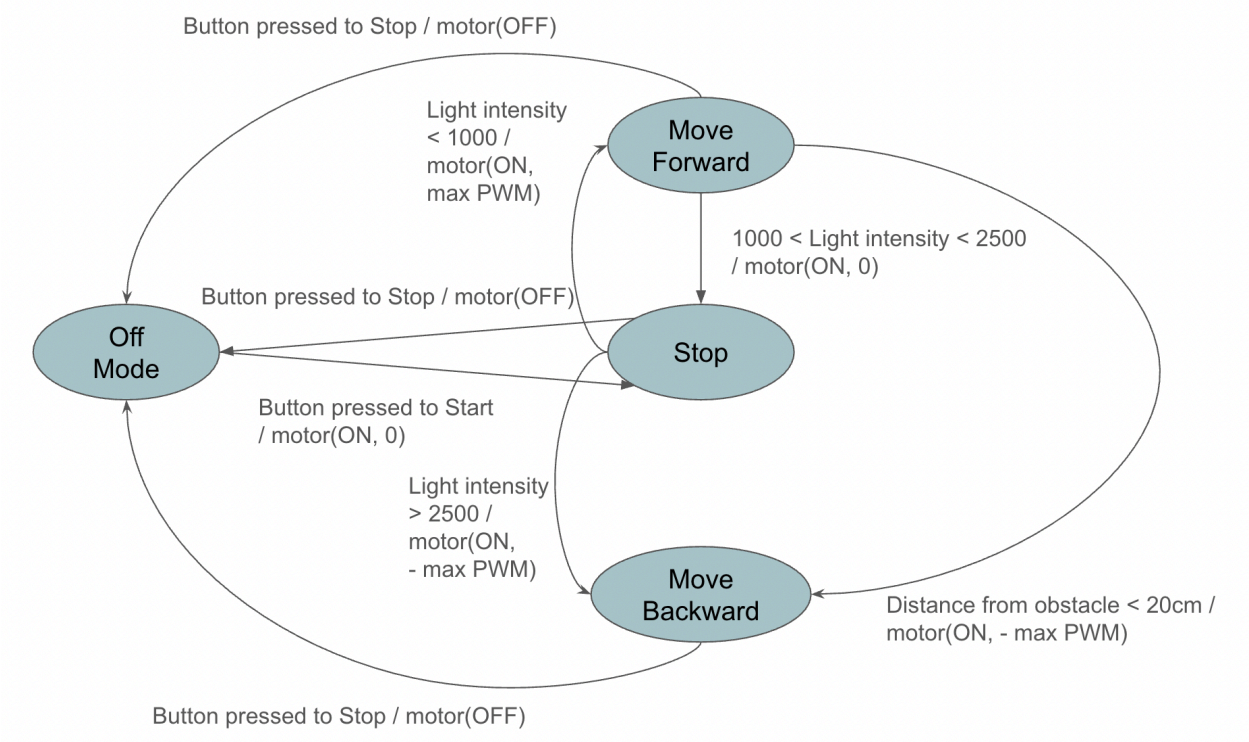
Torque_required = W_load * L_link / Mechanical Advantage
= 1 (kg) * 9.81 (m/s^2) * 0.1(m) / 4
= 0.245 (Nm)
= 24.5 (N*cm)

Using 2 motors on each side, dividing the torque required by 2,
We need 24.5/2 = 12.25 (N*cm) for each motor.
(Typical Theo Jansen linkage MA = 4)
Accordingly, we purchased 17.3 N*cm motors to account for added torque load from friction.

4. Circuit Diagram



5. State Diagram



6. Final Thoughts:

After many collaborative meetings throughout the semester, we successfully created a locomotive plant pot that could navigate homes to find sunlight. A key strategy that worked well for our group was active collaboration and open communication. By frequently sharing ideas and updating each other on progress, we were able to overcome challenges like optimizing the Theo Jansen leg design and balancing the robot's weight and motor torque. This collaborative approach kept everyone engaged and ensured all aspects of the project came together smoothly. Looking back, we wish we had allocated more time to implement additional features, such as a moisture sensor and automated watering system, which we had to shelve due to time constraints. Better time management and earlier prototyping could have helped us include these components. Aside from everything, we learned a lot from our mistakes in this project, but each one made us better engineers!

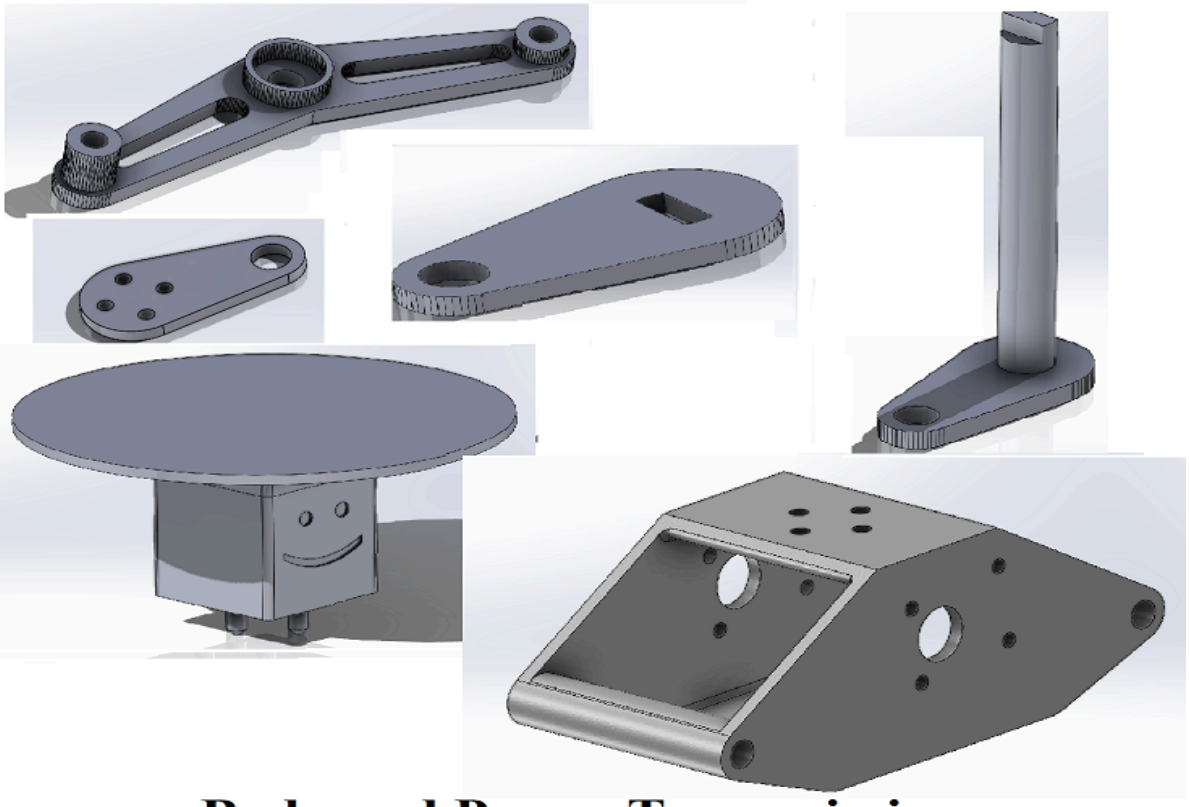
Appendix

Bill Of Materials

S-POT's Purchase Portfolio						$ 195.14
Component Name	**Listing Name**	**Quantity**	**Cost**	**Vendor**	**Link to Item**	**Subtotal**
DC Motors	BRINGSMART DC Motor 24V 260rpm DC Worm Gear Motor 6.2kg.cm Self-Locking Reversed Mini Turbine Geared Motor for DIY Robot Rotating Table Door Lock Curtain Machine (24V 260rpm)	2	$ 28.91	Amazon	https://www.amazon.com/gp/product/B07F8QG2LY/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1	$ 57.82
Screws	Screws	8	$ 0.18	Ace Hardware	No link	$ 1.44
Screws	Screws	8	$ 0.20	Ace Hardware	No link	$ 1.60
Nuts	Nuts	8	$ 0.16	Ace Hardware	No link	$ 1.28
Hex nuts	Hex Nuts	1	$ 10.99	Ace Hardware	No link	$ 10.99
Flat Washers	Flat Washers	1	$ 11.49	Ace Hardware	No link	$ 11.49
Threaded Rod	Threaded Rod 3/8" x 72"	2	$ 8.49	Ace Hardware	No link	$ 16.98
Power Supply	Power Supply Adapter, Signcomplex 24V 1A 12W Power AC Adapter Transformers, Switching Power Supply for LED Strip, Household Electronics, Output 24V DC, 5.5x2.1mm US Plug, UL Listed	1	$ 13.20	Amazon	https://www.amazon.com/dp/B0797MW8J6?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1	$ 13.20
Flange Coupling	4Pcs 8mm Flange Coupling Connector, Rigid Shaft Axis Fittings, Electroplated & High Hardness Heat-Treated for DIY RC Model Motors	1	$ 8.81	Amazon	https://www.amazon.com/dp/B0CSVZQHZY?ref=ppx_pop_dt_b_product_details&th=1	$ 8.81
H bridge	DROK DC Motor Driver, L298 Dual H Bridge Motor Speed Controller DC 6.5V-27V 7A PWM Motor Regulator Board 12V 24V Electric Motor Control Module Industrial 160W with Optocoupler Isolation	1	$ 15.99	Amazon	https://www.amazon.com/gp/product/B06XGD5SCB/ref=ppx_od_dt_b_asin_title_s00?ie=UTF8&psc=1	$ 15.99
Jacobs 3D Printing #1	pieza_g1_scaled1.6_15481041_15481070.gcode	1	$ 5.93	Jacobs 3D Printing	No link	$ 5.93
Jacobs 3D Printing #2	pieza_dbe1_scaled1.6_15301780.gcode	1	$ 2.98	Jacobs 3D Printing	No Link	$ 2.98
PLA+ Filament	2 kgs of PLA+ filament from Elegoo	1	$ 26.98	Amazon	https://www.amazon.com/dp/B0C14M5HR9/ref=twister_B0C14PXRZH?_encoding=UTF8&th=1	$ 26.98
Solar Panels	3Pcs Mini Solar Panel DC 6V Polysilicon Solar Cell Charger Module Solar DIY System Kits with 30cm Cable	1	$ 19.65	Amazon	https://drive.google.com/file/d/1vNG-_8mZriKwYQPo6O0A9vZgQFnN_uNc/view?usp=sharing	$ 19.65

CAD IMAGES

Leg Parts



Body and Power Transmission

SYSTEM CODE

```
#include <Arduino.h>
#define BTN 12      // Button pin
#define BIN_3 5
#define BIN_4 19

// Solar panel input pin
#define SOLAR_PIN 34

// Ultrasonic sensor pins
#define TRIG_PIN 7
#define ECHO_PIN 8

// LED pin (optional for status indication)
#define LED_PIN 13

// Setup variables ----------------------------------
const int freq = 25;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 250;
const int lowerThreshold = 1000;
const int higherThreshold = 2500;
const int obstacleDistanceThreshold = 20; // Distance in cm
unsigned long lastPressTime = 0;
const unsigned long debounceDelay = 200;
volatile bool buttonIsPressed = false;
int state = 0;

// Function prototypes
void moveForward();
void moveBackward();
void stopMotors();
bool isDistanceBeyondThreshold();
bool isLightIntensityBelowThreshold();
bool isLightIntensityAboveThreshold();
int readUltrasonicDistance();
int readSolarIntensity();
void handleState();
```

```
// Event checker functions
bool isDistanceBeyondThreshold() {
  int distance = readUltrasonicDistance();
  return distance > obstacleDistanceThreshold;
}

bool isLightIntensityBelowThreshold() {
  int intensity = readSolarIntensity();
  return intensity < lowerThreshold;
}

bool isLightIntensityAboveThreshold() {
  int intensity = readSolarIntensity();
  return intensity > higherThreshold;
}

// Service functions
int readUltrasonicDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  int distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);
  return distance;
}

int readSolarIntensity() {
  int intensity = analogRead(SOLAR_PIN);
  Serial.print("Solar Intensity: ");
  Serial.println(intensity);
  return intensity;
}

void IRAM_ATTR isr() {
  unsigned long currentTime = millis();
  if (currentTime - lastPressTime > debounceDelay) {
```

```
    buttonIsPressed = true;
    lastPressTime = currentTime;
  }
}

void setup() {
  // Serial communication for debugging
  Serial.begin(115200);


  ledcAttach(BIN_3, freq, resolution);
  ledcAttach(BIN_4, freq, resolution);

  // Solar panel pin setup
  pinMode(SOLAR_PIN, INPUT);

  // Ultrasonic sensor pins setup
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(BTN, INPUT);
  attachInterrupt(BTN, isr, RISING);

  // Set ADC resolution for ESP32
  analogReadResolution(12);
  analogSetAttenuation(ADC_11db); // Full 0-3.3V range
}

// Main loop -----------------------------------------
void loop() {
  if (buttonIsPressed) {
    buttonIsPressed = false;
    Serial.println("Button Pressed!");
    state = (state + 1) % 4; // Cycle through states: 0, 1, 2, 3
  }

  switch (state) {

    case 0: // Deactivated state
      Serial.println("State: Deactivated");
```

```
      stopMotors();
      break;

    case 1: // Moving forward
      Serial.println("State: Moving Forward");
      if (isDistanceBeyondThreshold()) {
        moveForward();
      } else {
        Serial.println("Obstacle detected! Stopping.");
        stopMotors();
      }
      break;

    case 2: // Stopping
      Serial.println("State: Stopping");
      stopMotors();
      break;

    case 3: // Moving backward
      Serial.println("State: Moving Backward");
      if (isDistanceBeyondThreshold()) {
        moveBackward();
      } else {
        Serial.println("Obstacle too close! Stopping.");
        stopMotors();
      }
      break;

    default:
      Serial.println("Invalid state. Resetting to deactivated.");
      state = 0;
      stopMotors();
      break;
  }

  delay(100); // Prevent CPU overload
}

// Functions ----------------------------------------
// Function to move forward
```

```
void moveForward() {
  //ledcWrite(BIN_1, 0);
  //ledcWrite(BIN_2, MAX_PWM_VOLTAGE);
  ledcWrite(BIN_3, 0);
  ledcWrite(BIN_4, MAX_PWM_VOLTAGE);
}

// Function to move backward
void moveBackward() {
  //ledcWrite(BIN_1, MAX_PWM_VOLTAGE);
  //ledcWrite(BIN_2, 0);
  ledcWrite(BIN_3, MAX_PWM_VOLTAGE);
  ledcWrite(BIN_4, 0);
}

// Function to stop all motors
void stopMotors() {
  //ledcWrite(BIN_1, 0);
  //ledcWrite(BIN_2, 0);
  ledcWrite(BIN_3, 0);
  ledcWrite(BIN_4, 0);
}
```