ME102B: Seed Planting Robot

ME 102B: Group 28: Charlize Niswanger, Sebastian Pescal, Jared Sitton

Opportunity

Planting seeds is a physically demanding task for individuals with personal gardens, especially the elderly, and those with back problems who endure significant physical strain. The seed-planting robot addresses these challenges by automating the seed planting process, including drilling holes and inserting seeds. Equipped with wheels and handles, the device allows users to easily move it to their desired location for planting seeds. This solution is especially beneficial for small-scale gardens or controlled environments, reducing manual effort and ensuring precise and efficient seed placement. Our initial design incorporated a rake to cover up the holes and a driving system, which we cut to reduce complexity. We had no desired speed for the system, only that the auger would be able to successfully dig through dirt and to not be burdened by the weight of itself, which we were able to accomplish.

High-Level Strategy

The seed-planting device is a user-operated tool with attached wheels for easy manual positioning, making it accessible and efficient for small-scale gardens. The planting process is initiated via a button press, which starts the automated seed planting sequence. The user can adjust the depth of the hole, ranging from 2 to 4 inches, using a potentiometer to accommodate different types of seeds. Users can fill the funnel with their seeds of choice, which will automatically fall into the seed sorter. Upon pressing the button, the auger will begin turning according to a depth determined by the potentiometer. To begin the auger will drill downwards, then return to its original position.

Once the auger completely finishes moving, the seed sorting system activates, separating the seeds one by one, and letting them fall down the tubing and into the hole drilled by the auger. The system operates with three key components: the opening, motor and cover discs. The opening disc aligns the isolated seed with the dispensing tube. The motor disc rotates by 60 degrees, controlled via an encoder, to position the seed slot over the opening. Finally, the cover disc prevents multiple seeds from being dispensed and protects the mechanism from potential jams. This mechatronic device has two degrees of freedom, with vertical and rotational motion, three DC motors, a button and potentiometer analog input, uses an ESP32 PICO, and operates by switching between states using events. This project uses elements from statics and materials as the basis of basic equations used to determine motors required and the project structure.

Function Critical Decisions

The auger system consists of three levels aligned by four shafts secured with shaft collars to ensure stability and precision during operation. The top and bottom plates align the lead screw and auger bit, while the middle plate connected to the lead screw nut enables controlled vertical motion. The system utilizes two motors to execute the drilling process. Motor 1, mounted to the middle plate, rotates the auger bit to drill into the soil. Motor 2, mounted to the top plate, rotates the lead screw, moving the middle plate vertically. To ensure smooth and reliable operation a flexible shaft coupling that transitions from 6mm to 10mm securely connects Motor 1 to the auger bit and Motor 2 to the lead screw. This setup ensures efficient drilling and accurate control over the hole depth for planting seeds.

Upon button activation, the auger system begins drilling a hole for a set duration based on the desired depth. The drill depth is scaled based on the potentiometer value using the equation:

$$\text{Hole Depth} = (P_{in} / P_{max}) * (4 \text{ in})$$

The drilling duration is calculated with the Motor 2's specified RPM, the pitch of the lead screw and the desired depth:

$$\text{Auger Duration} = (60\text{s}/251 \text{ rpm}) * (4 \text{ in}) / 0.0787 \text{ in/rev}$$

Utilizing the 12 V motor specifications and the auger dimensions we can check if it can accomplish the required torque to drill the hole on soft dirt. Soft dirt has a shear strength of approximately 25 kPa.

$$T_{drill} = \tau \cdot A \cdot r$$

$$r = \frac{diameter}{2} = \frac{13}{2} = 6.5\,mm = 0.0065\,m$$

$$A = \pi r^2 = \pi(0.0065)^2 = 1.33 \times 10^{-4} m^2$$

$$T_{drill} = 25000\,kPa \cdot 1.33 \times 10^{-4} m^2 \cdot 0.0065\,m$$

$$T_{drill} = 0.0216\,N \cdot m$$

The motor is capable of producing this torque without issue. The motor power output can also be checked at 130 rpm and $0.0216\,N \cdot m$

$$\omega = 130 \cdot \frac{2\pi}{60} = 13.61\,rad/s$$

$$P_{drill} = T_{drill} \cdot \omega = 0.0216 \cdot 13.61 = 0.294\,W$$

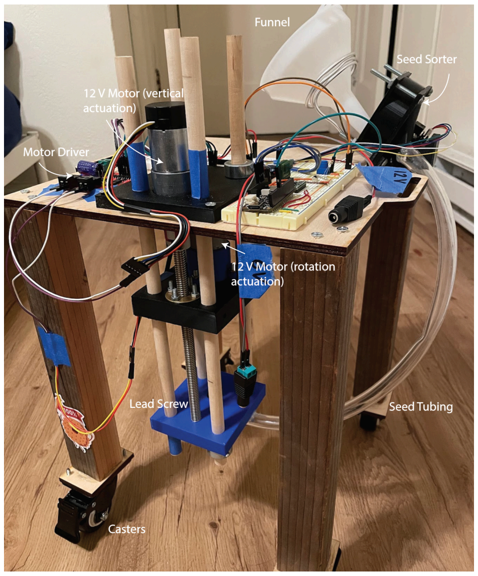
The motor can provide this power without problems. Finally operating can be analyzed as well.

$$Current = \frac{T_{drill}}{T_{stall}} \cdot I_{stall}$$

$$Current = \frac{0.0216}{2.156} \cdot 5 = 0.05\,A = 50\,mA$$

The motors current required is far below the rated limit of 5 A.

Integrated Physical Device


Funnel
Seed Sorter
12 V Motor (vertical actuation)
Motor Driver
12 V Motor (rotation actuation)
Lead Screw
Seed Tubing
Casters
Figure 1. Full Assembly

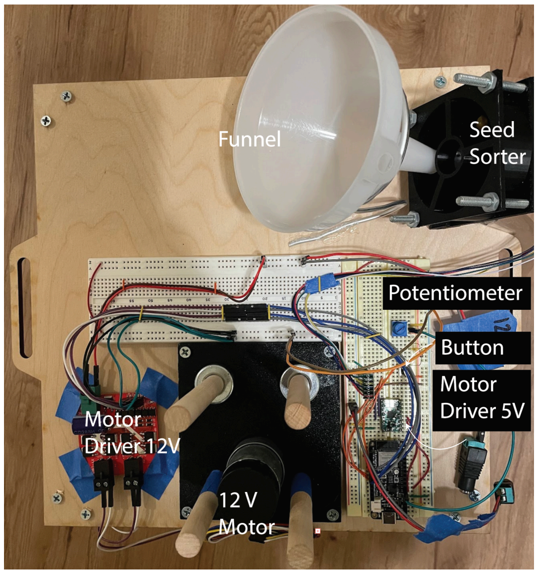


Figure 2. Top View

The above figures 1 & 2 depict the full assembly from a side and top down view. The listed components are the critical parts to keeping the system together and in motion. The funnel is held up by aluminum gage wire, and leads directly into the seed sorter, which is mounted on the edge of the table. The tubing runs from the seed sorter and directly down to the end of the auger and lead screw. The four wooden dowels surrounding the lead screw and auger keep the motion smooth and linear, and prevent wobbling as the system drills. The dowels and two 12V motors are mounted on a 3D printed, inset plane that is screwed into the table. The motor that spins the auger and the auger itself are connected to a platform that is connected to the lead screw, and this platform moves up and down as the lead screw turns.

The bottom blue 3D printed platform connects the dowels and the leadscrew, and the auger passes through a hole in the bottom to reach below. Two breadboards are used to secure the arduino, actuation button, potentiometer, the 5V motor driver, and many wires. The second breadboard leads the 12V power supply and connects the 12V motor driver to the ground, while also powering the 12V motors. The below figure shows a simplified model of the system without any wiring or breadboards, and helps visualize the mechanical parts of the system. The circuit diagram can be seen in figure 3.

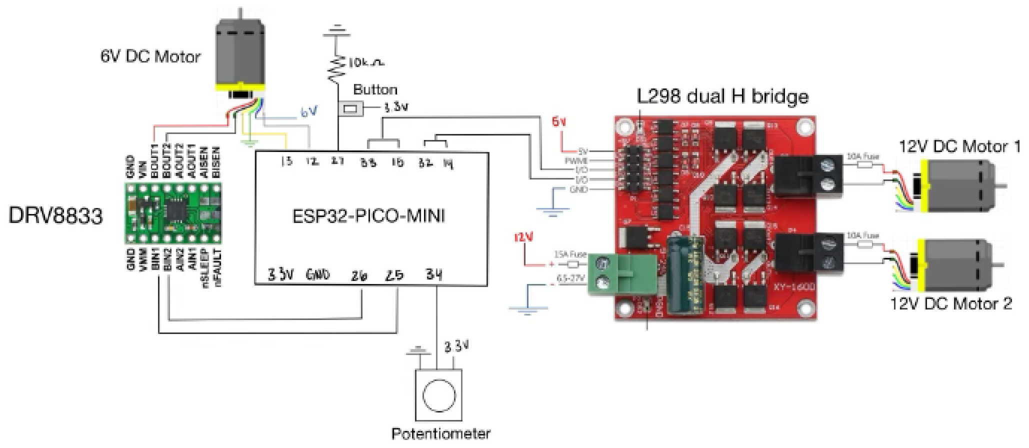


Figure 3. Circuit Diagram

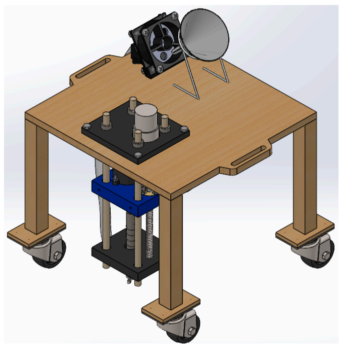


Figure 4. CAD Model

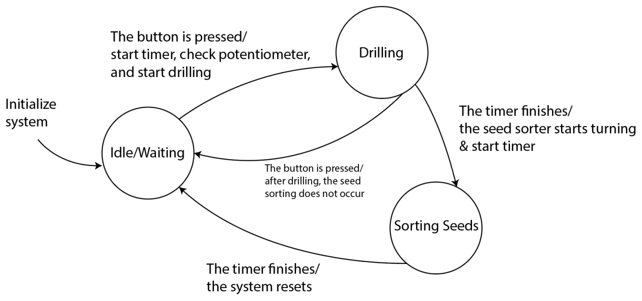

Initialize system
Idle/Waiting
Drilling
Sorting Seeds
The button is pressed/ start timer, check potentiometer, and start drilling
The timer finishes/ the seed sorter starts turning & start timer
The button is pressed/ after drilling, the seed sorting does not occur
The timer finishes/ the system resets

Figure 5. State Diagram

Figure 5 shows the simple state structure of the finished assembly. There are three main states- idling, drilling, and sorting. In the idling state, the machine is doing nothing and is simply awaiting input. When the button is pressed, a timer is started that is determined by the potentiometer level, and this timer will set how far and long the auger will drill for. This process cannot be interrupted, as we originally allowed for this, but it created issues in resetting the auger if it had not finished drilling or had not returned to its starting spot. However, if the button is pressed while the system is augering, after the drill resets the state will return to idling. After the auger returns to its original position, the seed sorter begins, which operates on a very brief timer. The seed sorter will first turn faster to overcome any slight friction caused by the close fit of parts. After the seed sorter finishes, the whole system returns to the idling state.

Reflection

Future students would benefit from getting the nuts and bolts details of their project flushed out early on, especially before the shop reflection. Having a rough SolidWorks mockup helps visualize where you want components to go, and what your overall system looks like, and also gives students a headstart on milestones. We were able to finish up the software and assembly of the system days in advance of the respective milestones, which was great for us being able to make small design tweaks and to reprint parts.

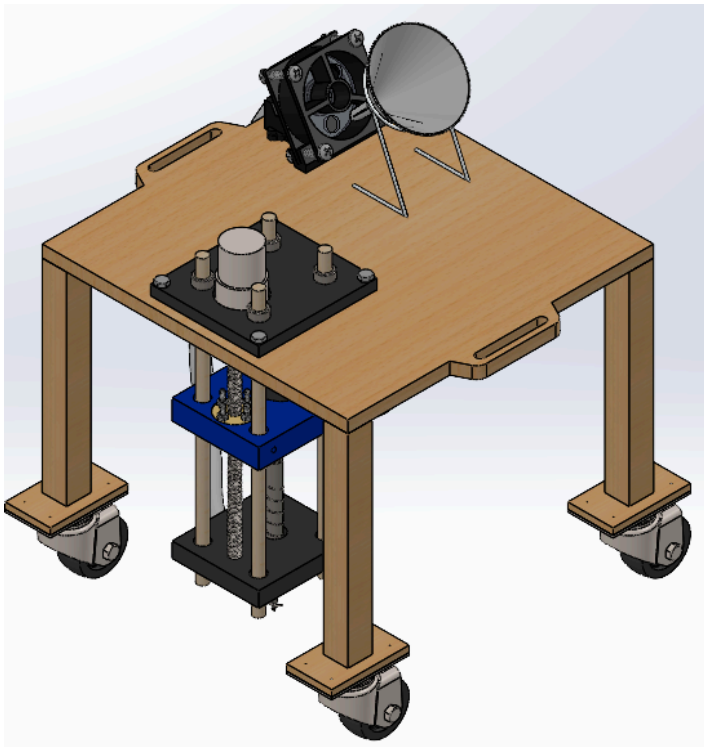
Appendix

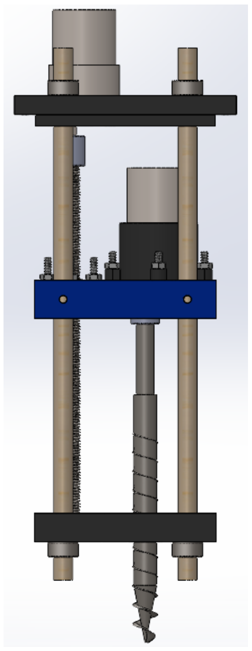
Bill of Materials

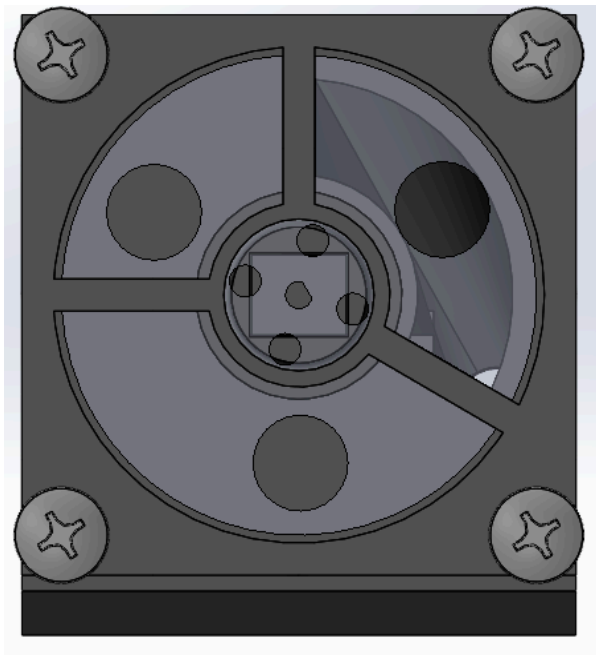
Components	Description	Quantity	Cost/Item	Total	Link
Base Plate	Base with electronics	1	Water Jet	0.00	N/A
Top Plate	Motor attachment and aligns rods	1	3D Print	0.00	N/A
Drill plate	Attached to lead screw enabling vertical motion and holds auger	1	3D Print	0.00	N/A
Bottom Plate	Align auger bit and rods	1	3D Print	0.00	N/A
Motor Mount - Drill plate	Attach motor to Drill plate	1	3D Print	0.00	N/A
Auger Bit	13mm Auger with 10mm attachment	1	Borrow	0.00	McMaster
Wooden Rods	1/2 in Dia.	4	1.75	6.99	Amazon
Flexible Shaft Coupling	6 mm to 10 mm shaft connection	2	9.99	19.98	Amazon
Lead screw and Nut	Controls vertical motion of auger	1	13.99	13.99	Amazon
Shaft Collar	Constrain 1/2 in rods	8	1.37	10.99	Amazon
12V Power Source	12V, 2A, Powers motors	1	8.98	8.98	Amazon
Gear motor (no encoder)	12 V, 130 RPM, 300 mA, 22 kg·cm (310 oz·in), 5A	2	Borrow(Tom)	0.00	Dfrobot
Gear motor w encoder	6 V, 410 RPM, 100 mA, 1.3 kg·cm (18 oz·in), 1.6 A	1	Borrow(Tom)	0.00	Pololu
Opening Disc	Housing component (Rigid) - 3D printed	1	3D Print	0.00	N/A
Motor Disc	Attached to Motor Shaft - 3D printed	1	3D Print	0.00	N/A
Housing Cover	Housing component (Rigid) - 3D printed	1	3D Print	0.00	N/A
Motor Mount	Attach motor to Baseplate at 30 degrees	1	3D Print	0.00	N/A
Set Screw Shaft Collars	Stabilize Motor Disc	1	2.00	2.00	McMaster
12V Power Source	12V, 2A, Powers motors	1	11.99	11.99	Amazon
2x2 Wooden Legs	Elevates baseplate	4	2.49	9.97	Home Depot
Casters	To mount on the legs to allow	4	3.37	13.49	Amazon

	movement of housing				
Plastic Funnel	Funnel to push seeds into seed sorter	1	5.53	5.53	Amazon
Wood Glue	Connect wood casters to legs of the housing	1	6.00	6.00	Ace Hardware
3 mm gage wire	Mount plastic Funnel on baseplate	1	12.99	12.99	Amazon
Clear tubing	To guide seed from sorter to hole	1	11.99	11.99	Amazon
3D Printer Filament	For 3D printed components	1	13.99	13.99	Amazon
M3 Pan Head Phillips Screws	M3 x 0.5 mm Thread, 15mm Long	8	0.35	2.8	Ace Hardware
M4 Pan Head Phillips Screws	M4 x 0.7mm Thread, 20mm Long	4	0.40	0.00	Ace Hardware
M4 Hex Nut	M4 x 0.7 mm Thread	4	0.33	1.32	Ace Hardware
M4 Pan Head Phillips Screws	M4 x 0.7mm Thread, 20mm Long	4	0.40	0.00	Ace Hardware
M4 Hex Nut	M4 x 0.7 mm Thread	4	0.33	1.32	Ace Hardware
M6 Pan Head Phillips Screws	M6 x 1mm Thread, 35mm Long for connecting opening disc, mount and housing cover	4	0.58	2.32	Ace Hardware
M6 Hex Nut	M6 x 1 mm Thread for connecting opening disc, mount and housing cover	4	0.35	1.40	Ace Hardware
No. 8 Wood Screw	No. 8 x 3/4 in, to attach components to baseboard	8	0.37	2.99	Ace Hardware
ESP32-PICO-Mini	Micro-controller	1	Lab Kit	0	N/A
DRV8833	2.7-10.8V 1.5A PWM motor driver board	1	Lab Kit	0	N/A
L298 Dual H Bridge Motor Speed Controller	6.5V-27V 7A PWM motor driver board	1	15.99	15.99	Amazon
			SUM:	177.02	

CAD Images


Full Isometric Assembly


Auger Transmission Subassembly


Seed Sorter Transmission Subassembly

Code

```
#include <Arduino.h>

#include <ESP32Encoder.h>

// Define Pins
#define LED_PIN 13
//Seed Motor & Encoder: Contorl Seed Sorter
#define BTN 27    // Button pin
//Auger Motor 1: Control Vertical Motion
#define BIN_1 33 // auger motor 1
#define BIN_2 15 // auger motor 1
//Auger Motor 2: Contol Drill
#define BIN_3 32 // auger motor 2
#define BIN_4 14 // auger motor 2
#define POT 34    // Potentiometer for controlling speed
//Seed Motor & Encoder: Contorl Seed Sorter
#define BIN_5 26 // seed motor
#define BIN_6 25 // seed motor
#define ENC_1 13 // Encoder input 1
#define ENC_2 12 // Encoder input 2

ESP32Encoder encoder;

// Setup variables ------------------------------------
// Setting PWM Properties
const int freq = 5000;            // PWM frequency
const int resolution = 8;         // PWM resolution (8-bit)
const int MAX_PWM_VOLTAGE = 255; // max PWM is 255

// State Variables
volatile bool sorterCheck = false;  // Check if seed sorter finished
bool augerStarted = false;          // Flag to track if auger is started
bool sorterStarted = false;         // Flag to track if sorter is started

// Encoder Variables
const int encoderCPR = 6;            // Encoder counts per revolution of
motor shaft
const float gearRatio = 75.81;       // Gear ratio of the motor
const float degreesToRotate = 120;   // Desired rotation in degrees
const float SeedM_PWM = 150;         // Maximum PWM value
// Motor Control Timer
volatile bool deltaT = false;        // check timer interrupt 2

// Computed Values: Encoder
float initialCount;
float targetCount;
float pulsesFor120Degrees;
float dynamicPWM;
float distRemaining;

// Button Variables
volatile bool buttonIsPressed = false;
volatile bool DEBOUNCINGflag = false;
hw_timer_t * buttonTimer = NULL;
portMUX_TYPE buttonTimerMux = portMUX_INITIALIZER_UNLOCKED;
// Debounce variables
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 1000; // 1s debounce delay

// Auger Variables
const int Time_Per_Rotation = 239; // Time for Auger Motor to complete 1
rotation in milliseconds (60/251)
unsigned long augerDuration = 0;   // Time for Desired Drill Depth

// Auger Variables
const float pitchDistance = 0.0787; // Vertical motion (inch) per
revolution
const float MIN_DrillDepth = 2;
const float MAX_DrillDepth = 4.0;
float holeDepth;

// POT Variables
const float MAX_POT = 4095;
float potValue;
float Scale;
int prevValue = 0;
int currentValue;

// System states
enum State { WAITING, SORTING, AUGERING };
State currentState = WAITING;

// Timer Variables
unsigned long startTime = 0;
unsigned long startTime_Seed = 0;
unsigned long lastCycleTimeA=0; // auger
unsigned long LastCycleTimeS=0; // sorter
unsigned long currentTime = 0;
unsigned long lastStepTime = 0;

// Initialization -----------------------------------
void IRAM_ATTR BTNisr() {  // Button press isr
  unsigned long currentTime = millis();
  if ((currentTime - lastDebounceTime) > debounceDelay) { // Check
debounce delay
    if (digitalRead(BTN) == HIGH) { // Ensure the button is pressed
      buttonIsPressed = true;
      timerStart(buttonTimer);
    }
    lastDebounceTime = currentTime; // Update last debounce time
  }
}
void IRAM_ATTR onTime() {
  portENTER_CRITICAL_ISR(&buttonTimerMux);
  DEBOUNCINGflag = false;
  portEXIT_CRITICAL_ISR(&buttonTimerMux);
  timerStop(buttonTimer);
}

void ButtonTimerInterruptInit() {
  // Initialize button debounce timer
  buttonTimer = timerBegin(80);   // timer 0, MWDT clock period = 12.5 ns
*
  // Attach the interrupt handler
  timerAttachInterrupt(buttonTimer, &onTime);
  // Stop the timer initially (debounce timer)
  timerStop(buttonTimer);
  timerAlarm(buttonTimer, 200000, true, 0); // 200000 * 1 us = 200 ms,
autoreload true
}

void setup() {
  Serial.begin(115200);
  //Setup Pin Modes
  pinMode(BTN, INPUT);      // Set button pin as input
  pinMode(POT, INPUT);      // Set Potentiometer as input
  pinMode(BIN_1, OUTPUT);   // Motor control pin for auger 1
  pinMode(BIN_2, OUTPUT);   // Motor control pin for auger 1
  pinMode(BIN_3, OUTPUT);   // Motor control pin for auger 2
  pinMode(BIN_4, OUTPUT);   // Motor control pin for auger 2
  pinMode(BIN_5, OUTPUT);   // Motor control pin for seed motor
  pinMode(BIN_6, OUTPUT);   // Motor control pin for seed motor

  // Encoder Setup
  ESP32Encoder::useInternalWeakPullResistors = puType::up;  // Enable the
weak pull up resistors
  encoder.attachHalfQuad(ENC_1, ENC_2);                     // Attache
pins for use as encoder pins
  encoder.setCount(0);                                      // set
starting count value after attaching
  initialCount = 0;
  pulsesFor120Degrees = encoderCPR * gearRatio * degreesToRotate / 360;

  // Button Setup
  attachInterrupt(BTN, BTNisr, RISING);  // Interrupt on button press
  ButtonTimerInterruptInit();            // Initialize timer for button
debounce

  // Motor Setup
  // configure PWM functionalities with channels to the GPIO to be
controller
  ledcAttach(BIN_1, freq, resolution);
  ledcAttach(BIN_2, freq, resolution);
  ledcAttach(BIN_3, freq, resolution);
  ledcAttach(BIN_4, freq, resolution);
  ledcAttach(BIN_5, freq, resolution);
  ledcAttach(BIN_6, freq, resolution);
}

// Main loop -----------------------------------
```

```
void loop() {
  // Check Potentiometer
  currentValue = analogRead(POT);
  if (abs(currentValue - prevValue) > 100) {
    // Set Drill Depth
    drillDepth();
    Serial.print("Drill Depth: ");
    Serial.println(holeDepth + MIN_DrillDepth); // Print the Drill Depth
    prevValue = currentValue;                   // Update the previous
value
  }

  // EVENT CHECKER
  if (CheckForButtonPress()) {
    ButtonResponse();  // Handle button press
  }

  // Act based on the current state
  switch (currentState) {
    case WAITING:
      // Wait for the button press to start the process
      ledcWrite(BIN_1, LOW);
      ledcWrite(BIN_2, LOW);
      ledcWrite(BIN_3, LOW);
      ledcWrite(BIN_4, LOW);
      ledcWrite(BIN_5, LOW);
      ledcWrite(BIN_6, LOW);
      break;

    case AUGERING:
      if (augerStarted) {
        turnAuger(); // SERVICE FUNCTION
      }
      break;

    case SORTING:
      // Start seed sorting process
      Serial.println("In sorting state");
      turnSeedSorter(); // SERVICE FUNCTION
      break;
   }
}

void drillDepth() {
  // Pot Setup
  potValue = analogRead(POT);
  Scale = potValue / MAX_POT;
  // Auger Setup
  holeDepth = Scale * (MAX_DrillDepth - MIN_DrillDepth);
  augerDuration = Time_Per_Rotation * (MIN_DrillDepth + holeDepth) /
pitchDistance;
}

// Call this function to start the auger
```

```
void startAuger() {
  startTime = millis();  // Capture the start time
  augerStarted = true;   // Set the flag to start the auger
  Serial.println("Auger started.");
}

// Turn the auger motor ------------------------------------
void turnAuger() {
  if (augerStarted) {  // Check if the auger should run
    unsigned long currentTime = millis();  // Get the current time
    Serial.println("The auger is now turning for " +
String(2*augerDuration/1000) + " seconds");
    while (currentTime - startTime < augerDuration) {
      currentTime = millis();   // Get the current time
      // Set Auger 1 Motor Speed - CCW
      ledcWrite(BIN_1, LOW);
      ledcWrite(BIN_2, MAX_PWM_VOLTAGE);
      // Set Auger 2 Motor Speed - CCW
      ledcWrite(BIN_3, LOW);
      ledcWrite(BIN_4, MAX_PWM_VOLTAGE);
    }
    while (currentTime - startTime < (2.00 * augerDuration)) {
      currentTime = millis();   // Get the current time
      //Set Auger 1 Motor Speed - CW
      ledcWrite(BIN_1, MAX_PWM_VOLTAGE);
      ledcWrite(BIN_2, LOW);
      //Leave Auger 2 Motor Speed - CCW
    }
    // After augerDuration, stop the motor and reset the augerStarted flag
    augerStarted = false;  // Stop the auger
    Serial.println("Auger operation completed.");
    // Stop all auger motors
    ledcWrite(BIN_1, LOW);
    ledcWrite(BIN_4, LOW);
    ledcWrite(BIN_4, 0);
    ledcWrite(BIN_1, 0);

    startSorter();            // start seed sorter
    currentState = SORTING;   // Transition to sorting state after auger is
done
  } else {
    Serial.println("No auger start signal.");
  }
}

void startSorter() {
  startTime_Seed = millis();  // Capture the start time
  sorterStarted = true;  // Set the flag to start the sorter
  Serial.println("Sorter started.");
}

// Turn the seed sorter motor ------------------------------------
void turnSeedSorter() {
  // Stop all auger motors
  ledcWrite(BIN_1, LOW);
  ledcWrite(BIN_2, LOW);
  ledcWrite(BIN_3, LOW);
  ledcWrite(BIN_4, LOW);
  // Update Target
  targetCount = initialCount + pulsesFor120Degrees;
  // Rotate until target count is reached
  ledcWrite(BIN_5, LOW);
  while (encoder.getCount() < targetCount) {
    // Set motor direction to forward
    distRemaining = targetCount - encoder.getCount();
    dynamicPWM = map(distRemaining, 0, pulsesFor120Degrees, 80,
SeedM_PWM);
    ledcWrite(BIN_6, dynamicPWM);

    // Account For Encoder Update Speed
    if (distRemaining < 4) {
      Serial.println("Stop Range Reached");
      ledcWrite(BIN_6, 0);
      break;
    }
  }

  // Stop the motor
  ledcWrite(BIN_5, 0);
  ledcWrite(BIN_6, 0);

  Serial.println("Target reached.");
  initialCount = targetCount;

  Serial.println("Seed sorter finished.");
  currentState = WAITING;  // Transition to sorting state after sorter is
done
  sorterStarted = false;   // Reset Values
}

// Button press handler ------------------------------------
bool CheckForButtonPress() {
  if (buttonIsPressed && !DEBOUNCINGflag) {
    portENTER_CRITICAL_ISR(&buttonTimerMux);
    DEBOUNCINGflag = true;  // Set debounce flag
    portEXIT_CRITICAL_ISR(&buttonTimerMux);
    timerRestart(buttonTimer);  // Restart the debounce timer
    return true;
  } else {
    return false;
  }
}

// Button reset ------------------------------------
void ButtonResponse() {
  // If we're already in the middle of a process, reset everything
  if (currentState != WAITING) {
    Serial.println("Button pressed again! Resetting process...");
    augerStarted = false;
    sorterStarted = false;
    currentState = WAITING;  // Return to the waiting state
    // Stop all motors
    ledcWrite(BIN_1, LOW);
    ledcWrite(BIN_2, LOW);
    ledcWrite(BIN_3, LOW);
    ledcWrite(BIN_4, LOW);
    ledcWrite(BIN_5, LOW);
    ledcWrite(BIN_6, LOW);

  } else {
    // If in WAITING state, start augering process
    Serial.println("Button pressed! Auger starting...");
    startAuger();
    currentState = AUGERING;  // Move to auger state
    augerStarted = true;      // Start auger
  }

  buttonIsPressed = false;  // Reset button press flag
  portENTER_CRITICAL_ISR(&buttonTimerMux);
  DEBOUNCINGflag = false;   // Reset debounce flag
  portEXIT_CRITICAL_ISR(&buttonTimerMux);
}
```

Video Link:

Group 28 Seed Planter.mp4