ME102B - Assistive Rotating Toolbox

Nolan Herbert, Henry Libermann, Vic May

December 19, 2024

Opportunity

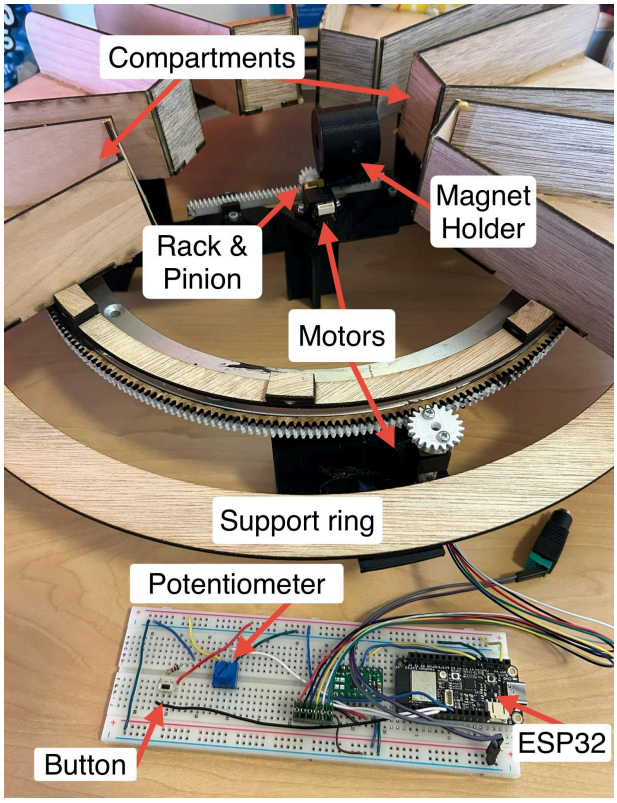
Often when building you will find yourself in need of a tool that you can't seem to find. That's where the assistive rotating toolbox comes in. By simply controlling the toolbox to rotate the correct compartment to the front then pressing a button, the tool you need will be pushed out for you to grab, use, then place back in the toolbox. This allows you to stay organized and efficient while working without having to pile your tools on the ground next to you or run back and forth to the toolbox when something you need wasn't grabbed the first time.

High Level Strategy

1.) Select amount of items and place them in the desired compartment accordingly
2.) Turn the dial to bring the desired item to the intended position
3.) Press the button to push the item compartment out
4.) Remove the desired item

The overall structure has not changed since our original high level strategy was Introduced, but some significant changes have been made. Where we originally intended for the selection to be voice activated, we instead opted for the rotation of the platform housing the compartments to be operated by hand. We did so to simplify the code and make integration a much simpler task while still demonstrating the same functionality. The second major change was the inclusion of the button. Originally we intended for the compartments to move to the desired location and then push the compartment out in one fluid motion, but the final product had the pushing of the compartment operated by a button press. This was done to give the user more control over the timing of the item release. These changes did alter the use case of the item. What was originally intended to be hands free now requires physical inputs, which means that the original goal of continuing work while tools are delivered to the user had to be altered.

Photos of Device



Function Critical Decisions

We elected to drive the rotational element of our toolbox using gears, in order to maximize torque and prevent slippage between the lazy susan bearing and rotational motor. The same motors (provided in our lab kits) were used for the pushing/pulling mechanism and the rotational driver to simplify wiring and use a single ESP32. The relatively lightweight materials of laser cut wood and PLA were substituted for heavier prototyping materials to reduce the necessary force applied for rotation.

Related Calculations:

Estimated load: 700 g = 0.7 kg

r = 6 in = 0.1524 m

Angular velocity of motor (omega): 8 counts per revolution

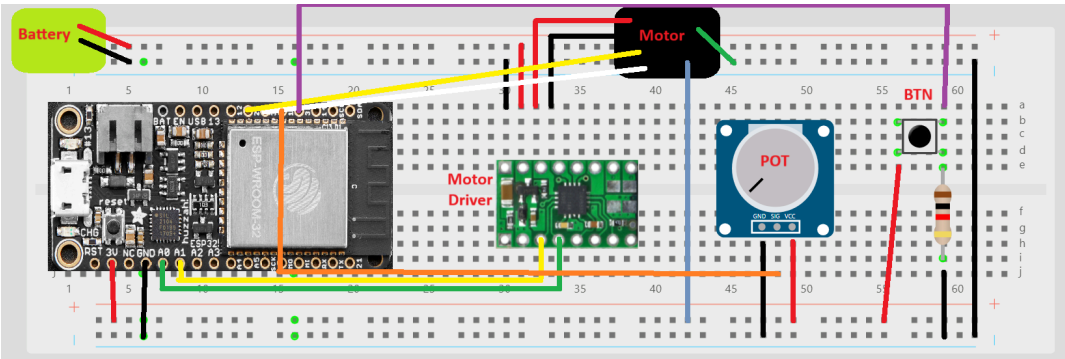
Estimated force applied in the z-axis (Fz): $0.7\ kg\ \cdot\ 9.81\ m/s^2\ =\ 6.867\ (kg\ \cdot\ m)/s^2$

Estimated force applied in the x-y plane (Fxy): 8 counts per rev * 0.7 kg = 5.6 kg*counts per rev

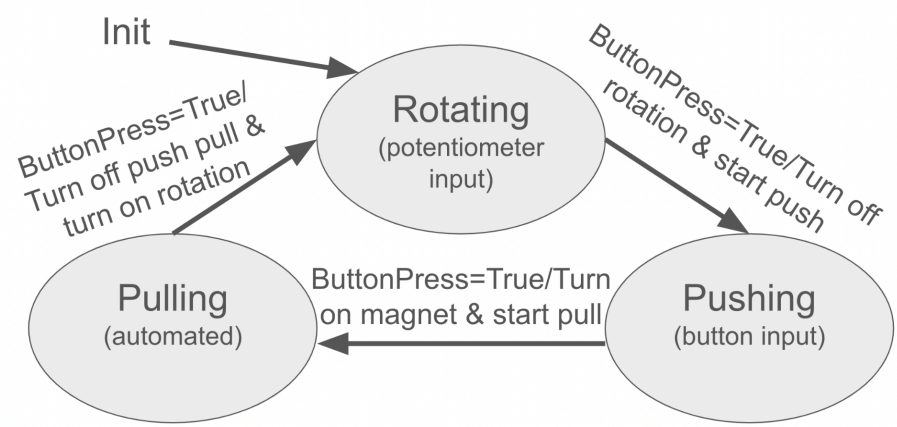
Torque: $\tau\ =\ rF\ =$ 0.85344 kg*m*counts per rev

Gear ratio: 12 in / 1 in = 12

Circuit & State Transition Diagrams



Above: Circuit diagram of final setup



Above: State Transition Diagram

Reflection

Overall, the members of our team agree that the execution of this project could have been much better. Although we were able to achieve some level of success in regards to basic functionality and part design, the final product was plagued with issues that could have easily been avoided. Our group's greatest strength was creating the initial concept and quickly identifying necessary components. We were able to quickly acquire all off the shelf hardware long before we were able to assemble our prototype. Our biggest challenge was integrating all of our hardware in a timely fashion. Unfortunately, we did run into some unanticipated issues that greatly set us behind schedule, but this could have been avoided if we began to manufacture our parts and test fit everything even just one week sooner. We were forced to assemble everything at the last minute, which left us with no time to test and troubleshoot our product. This resulted in a project that technically functioned and met minimum requirements, but lacked a sense of refinement. As aspiring engineers we seek to produce great designs that excel at their desired function, and what we produced just didn't meet those expectations. However, we know exactly what we need to fix and what we could have done better. Our biggest takeaway from this is to not underestimate just how meticulous the process of integration and manufacturing is. Both processes are time consuming and critical to creating a reliable project.

Video: 102B.mov

Appendix

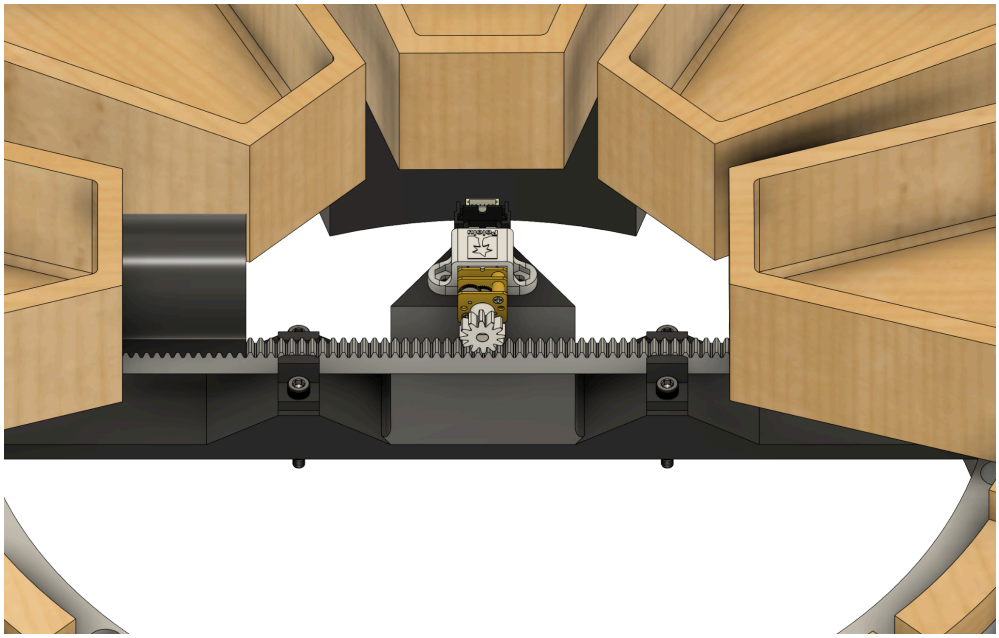
Bill Of Materials

Total Cost (w/o tax)	$72.02	(Excluding Rejected)	Total Spent (w/o tax)	$72.02	(Ordered and Arrived)		ASME Grant Amount		$156.03	
Item	**Comments**	**Status**	**Who added it**	**Listed Name/ Product Number**	**Link**	**Vendor**	**Cost/ Item**	**Order Quantity**	**Cost**	**Who's Buying/Making**
Lazy Susan Bearing	12", Possibly going to get one from goodwill	Arrived	Nolan	TamBee 12 Inch Lazy Susan Hardware Heavy Duty Metal Rotating Hardware Turntable Bearing	Link	Amazon	$22.99	1	$22.99	Nolan
Gear Rack	In place of linear actuator, requires gear and motor (will have to cut down to size)	Arrived	Nolan	20 Degree Pressure Angle Gear Rack, 0.8 Module / 2662N56	Link	McMaster-Carr	$3.80	2	$7.60	Nolan
Pinion	Drives rack	Arrived	Nolan	20 Degree Pressure Angle Plastic Gear /	Link	McMaster-Carr	$5.72	2	$11.44	Nolan

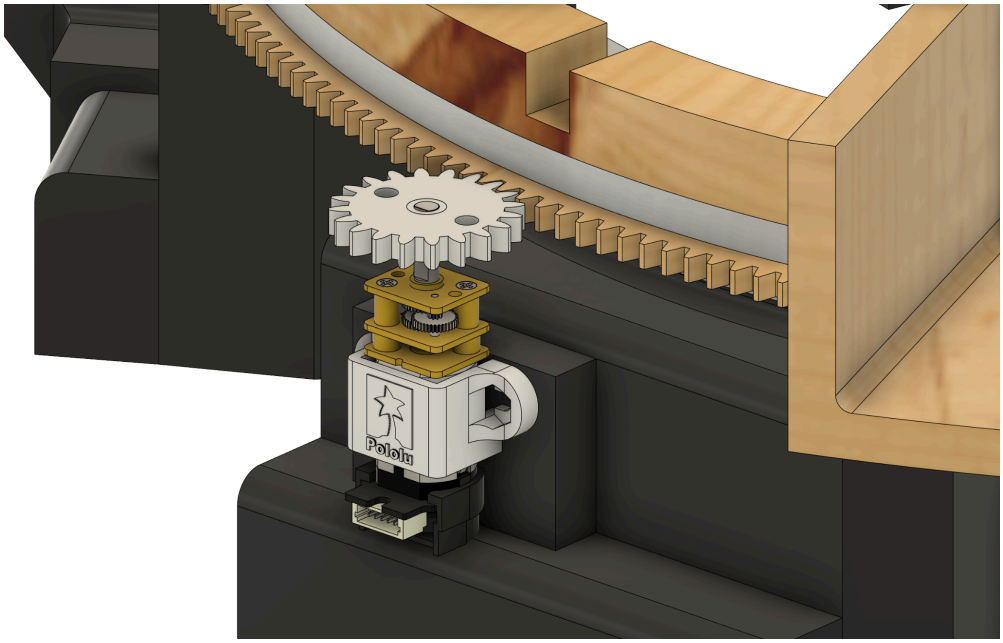
				2662N322						
Motors	Lab Kit motors are free	Arrived	Nolan			Lab Kit	$0.00	2	$0.00	All
Motor bracket	In lab kit	Arrived	Nolan			Lab Kit	$0.00	2	$0.00	All
Bolts	M3, for gear rack attachment (25 per package)	Arrived	Nolan	Zinc-Flake-Coated Alloy Steel Socket Head Screw / 91274A111	Link	McMaster-Carr	$3.95	1	$3.95	Nolan
Nuts	M3 (100 per package)	Arrived	Nolan	Steel Hex Nut / 90592A009	Link	McMaster-Carr	$2.23	1	$2.23	Nolan
Plastite Screw	M3 (for electromagnet to rack connection, pack of 50)	Arrived	Nolan	Stainless Steel Flat Head Thread-Forming Screws for Plastic / 90485A445	Link	McMaster-Carr	$7.26	1	$7.26	Nolan
Electromagnet	Used to pull back empty compartment	Arrived	Nolan	5V ELECTROMAGNET 2.5 KG HOLDING	Link	DigiKey	$7.50	1	$7.50	Nolan
Rack Supports	Custom Made (Cost included in	Arrived	Nolan					4	$0.00	Nolan

	magnet holder cost)									
Electromagnet Holder	Custom Made	Arrived	Nolan				$1.76	1	$1.76	Nolan
Leg Supports	Custom Made (set of 3)	Arrived	Nolan				$7.29	1	$7.29	Nolan
Central Support	Custom Made	Arrived	Nolan				$6.23	1	$6.23	Henry
Center Support Leg	Custom Made	Arrived	Henry				$1.29	1	$1.29	Henry
Compartments	Custom Made	Arrived	Nolan				$0.68	8	$5.44	Henry/ Vic
Compartment Holder	Custom Made	Arrived	Nolan				$5.44	1	$5.44	Henry
Outer Gear	Custom Made	Arrived	Henry				$7.12	1	$7.12	Henry
Outer Motor Gear	Custom Made	Arrived	Henry				$1.07	1	$1.07	Henry
Support Ring	Custom Made	Arrived	Henry				$2.72	1	$2.72	Henry
Support Ring Legs	Custom Made	Arrived	Henry				$4.62	1	$4.62	Henry

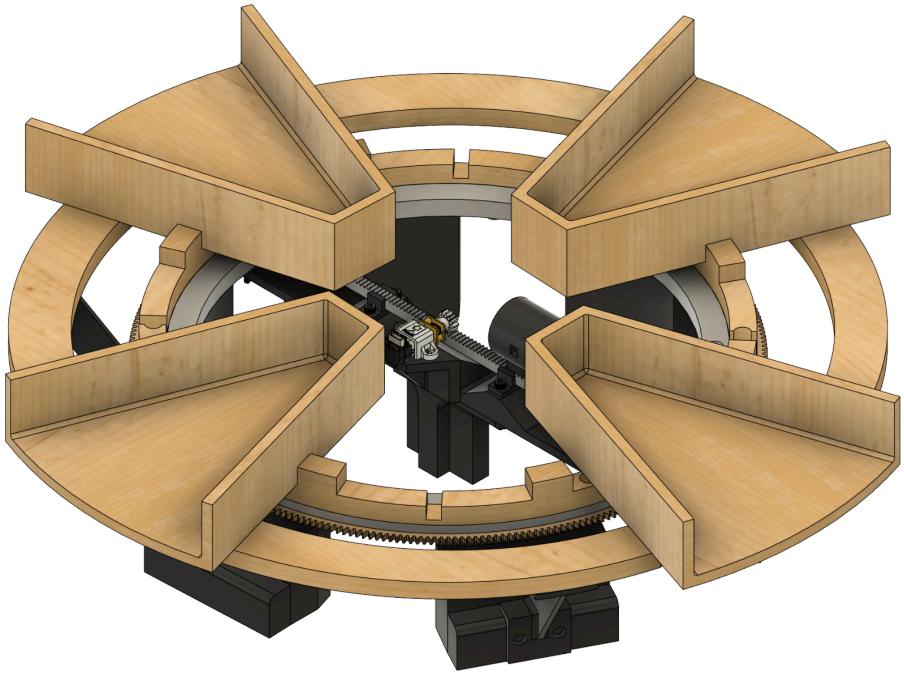
CAD



Rack and pinion linear motion mechanism



Outer ring gear rotation mechanism



Isometric View

Code

```
#include <Arduino.h>
#include <ESP32Encoder.h>
#define BTN 32  // declare the button ED pin number
#define BIN_1 26
#define BIN_2 25
#define POT 15
#define BIN_3 4

byte state = 0;
ESP32Encoder encoder;

int theta = 0;
int thetaDes = 0;
int thetaMax = 455;  // 75.8 * 8 counts per revolution
int D = 0;
int potReading = 0;
int err = 0;
int err_sum = 0;

int Kp = 4;  // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
```

```
int Ki = 0.095;
int IMax = 150;

//Setup variables
volatile bool buttonPressed = false;
volatile bool DebouncingFlag = false;

//Setup interrupt variables
volatile int counter = 0; // encoder count

//volatile bool interruptCount = false; // check timer interrupt
//int totalInterrupts = 0; // # of interrupts triggered

hw_timer_t*timer0 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties ----------------------------
const int freq = 5000;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 100;
int v = 0;

const int ledChannel_1 = 1;
const int ledChannel_2 = 2;

void IRAM_ATTR isr() {  // the function to be called when interrupt is
triggered
  buttonPressed = true;
}

void IRAM_ATTR onTime0(){
  portENTER_CRITICAL_ISR(&timerMux0);
  DebouncingFlag = false;
  //interruptCount = true; // func being called at interrupt
  portEXIT_CRITICAL_ISR(&timerMux0);
  timerStop(timer0);
  buttonPressed = false;
}
```

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(BTN, INPUT);                // configures the specified pin to
behave either as an input or an output
  attachInterrupt(BTN, isr, RISING);  // set the "BTN" pin as the
interrupt pin; call function named "isr" when the interrupt is triggered;
"Rising" means triggering interrupt when the pin goes from LOW to HIGH

  timer0 = timerBegin(100000); // frequency = 1Mhz
  timerAttachInterrupt(timer0, &onTime0);
  timerAlarm(timer0,25000, true, 0);

  pinMode(POT, INPUT);
  ESP32Encoder::useInternalWeakPullResistors = puType::up;  // Enable the
weak pull up resistors
  encoder.attachHalfQuad(27, 33);                           // Attache
pins for use as encoder pins
  encoder.setCount(0);                                      // set
starting count value after attaching
  // configure PWM functionalitites with attaching the channel to the GPIO
to be controller
  ledcAttach(BIN_1, freq, resolution);
  ledcAttach(BIN_2, freq, resolution);

  pinMode(BIN_3, OUTPUT);
  digitalWrite(BIN_3, LOW);
}

void loop() {
  // put your main code here, to run repeatedly:
  theta += -counter;
  potReading = analogRead(POT);
  thetaDes = map(potReading, 0, 4095, 0, thetaMax)*(30/1.75); //scaled for
gear ratio

  err = thetaDes-theta;
  err_sum = err_sum + (err / 10);
  if(err_sum > IMax){
    err_sum = IMax;
```

```
}
D = Kp * err + Ki*err_sum;

//Ensure that you don't go past the maximum possible command
if (D > MAX_PWM_VOLTAGE) {
  D = MAX_PWM_VOLTAGE;
  err_sum = err_sum - (err / 10);
} else if (D < -MAX_PWM_VOLTAGE) {
  D = -MAX_PWM_VOLTAGE;
  err_sum = err_sum - (err / 10);
}

if (D > 0) {
  ledcWrite(BIN_1, LOW);
  ledcWrite(BIN_2, D);
} else if (D < 0) {
  ledcWrite(BIN_2, LOW);
  ledcWrite(BIN_1, -D);
} else {
  ledcWrite(BIN_2, LOW);
  ledcWrite(BIN_1, LOW);
}
//plotControlData();
Serial.println("Bed Rotating");
delay(100);

switch (state) {
  case 0 : // OFF

    delay(1000); //1 sec of blocking code

    Serial.println("Slow down buddy!");

    state = 1;
    break;

case 1 : // ON

  if(CheckForPress()){
    ButtonResponse();
```

```
      //Serial.println("Arm Extended & Electromagnet Engaged!");
      state = 0;
    }
    break;

  default: //Err
    Serial.println("SM_ERROR");
    break;

  }
}



bool CheckForPress(){
  if(buttonPressed == true && DebouncingFlag == false){
    portENTER_CRITICAL_ISR(&timerMux0);
    DebouncingFlag = true;
    portEXIT_CRITICAL_ISR(&timerMux0);
    timerStart(timer0);
    return true;
  }
  else {
    return false;
  }
}

void ButtonResponse(){
  buttonPressed = false;
  Serial.println("Arm Extended & Electromagnet Engaged!");
  ledcWrite(BIN_1, LOW);
  ledcWrite(BIN_2, 150);
  ledcWrite(BIN_3, HIGH);
  delay(600);
  ledcWrite(BIN_2, LOW);
  ledcWrite(BIN_1, 150);
  ledcWrite(BIN_3, HIGH);
  delay(600);
  ledcWrite(BIN_2, 0);
  ledcWrite(BIN_1, 0);
```

```
  ledcWrite(BIN_3, LOW);
}
```