ME102B Final Report

James Lee, Hae Young Jang, Dorotea Macri

Opportunities

The automated telescope adjuster creates the opportunity and accessibility to tools, skills, and instruments (specifically telescopes) that are inaccessible to many due to the lack of technical skill and knowledge to use them. The project is intended to act as 'training wheels' to help amateur astronomers learn to position a telescope. The difficulty of accurately aiming a telescope is a major hurdle to many aspiring amateur astronomers, since it is a challenging skill to learn. Our project aims to help users overcome this barrier to entry by providing feedback and adjustments to users' attempts to position the telescope themselves.

Device High Level Strategy

In order to begin using the telescope, a user will first press the motor power button in order to activate the automatic telescope and its motors. The user will then move the telescope manually to the general area of a star, planet, or desired location. The telescope contains angular location information of various planets (the moon, Venus, etc.) stored from previous testing with a 6-dof gyroscope input sensor. Once the user finishes moving the telescope and a location has been chosen, the user will then press the automation state button in order to actuate the telescope automatically to get a more accurate and focused view of the desired location or object.

After the automation state has been activated, the telescope compares its current angular location, from the input gyroscope sensor, to the stored locations of various planets and the telescope will then begin to actuate to get a view of the closest object in its viewing vicinity. It first actuates in the azimuth direction with stepper motor 1. When finished in the azimuth direction, the telescope actuates in the elevation direction with stepper motor 2 to finalize its view of the desired object.

Once the viewing position has been finalized, the telescope exits the automation state and becomes idle. It then prints an information block of the viewed object to the serial monitor so that the user can learn more about what they are viewing. After the user finishes viewing, they can repeat the steps above in order to view another object.

Special considerations that may be included in the future include increasing the mobility of the telescope to consider conditions like soft ground and strong wind in order to create accurate and focused images. Another is creating a stronger frame to accommodate larger telescopes, or additional UIs for better educational purposes.

Initial desired functionality vs achieved specifications

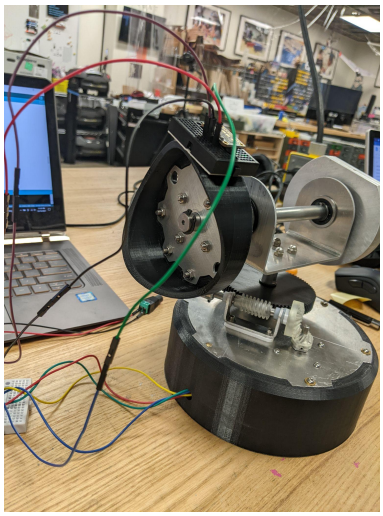
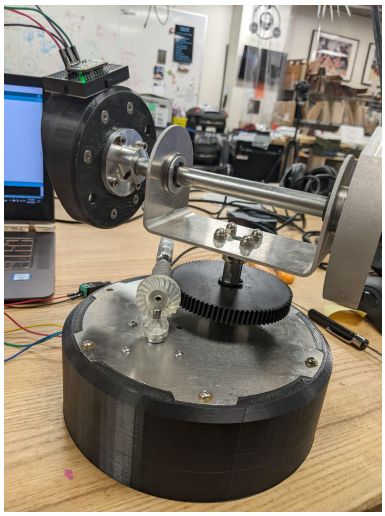
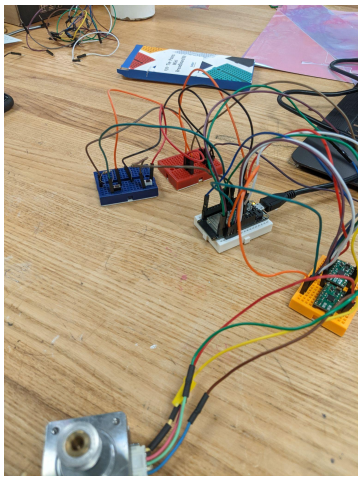
Initially the only functionality we were targeting was accurate actuation in the azimuth and elevation directions. Our goal was to provide positional accuracy of at least an arcminute, which is one-sixtieth of a degree. We used gears of a ratio of 540 to provide accuracy at an arcminute.

Although our stepper motors can comfortably run at 600 RPM, we did not need to reach any specific high motor speeds as time to reach positions was not a specification we were targeting. We drove our motors at 10 RPM to maintain the holding torque of the stepper motors and move at a reasonable rate. There was also no need for either axis to ever move more than half a rotation about an axis at a time as most movements performed were much smaller.

Also we chose to use a gyro sensor to detect telescope motion while a human moves it and turns into a moving state automatically. However, we realized that sometimes one needs to move the

telescope very slowly to achieve desired state, or even want to stop moving for a while and then try moving it again. Considering all various situations, we decided to implement button technique to get into automation state instead of gyro sensor reading.

System Photo

Function Critical Decisions and Calculations

Bearings

Many design choices were made to be over-robust, since our speeds and loads were reasonably low, but the use case-- a device that can be moved around, ideally take some damage and not stop working-- made designing for strength desirable. We chose a set of bearings intended for lawnmowers and wheelbarrows, which would subject them to much heavier loads and more wear than our device. They are also sealed, which is preferable for a device being used outside.

Gear Ratio and Worm Drive

Our motors provide 200 full steps per rotation, or 400 half-steps. Our goal is to provide position accuracy of at least an arcminute, which is one-sixtieth of a degree. This means we need a gear ratio of at least 54, although a ratio of 540 or 5400 would be preferable. To achieve this, we chose to use a worm drive, since this gearing scheme allows for a very large gear ratio in a small form factor.

Based on some research, large worm gears are fairly common for astronomy applications. However, they are typically homemade using a very unconventional lathe setup. Worm gears of this type are not typically available from suppliers at affordable price points. As an alternative, we chose to 3D print the main worm gear using carbon fiber and reinforced plastic. The speeds we are running at are low, we did not require too much strength from the gear, and lower weights are more desirable on gears. We realize this is not an ideal manufacturing choice, it seemed to be the best choice given the constraints.

Bevel Gears

The use of bevel gears allowed us to conceal and protect the motor and wires below the surface of our hardware enclosure. They also allowed us to double our overall gear ratio, meaning we saved substantial space in our worm gear scheme. Finally, it was desirable not to have the motor directly driving the output shaft in case of any displacement or sudden loads on the shaft that might damage the motor. The bevel gears are made of cast plastic with aluminum cores. The use of plastic and aluminum for most of the ‘external’ parts of the machine was to avoid corrosion in case the device got left outside (as is common with telescopes).

Stepper Motors

Our application requires precise position control in 2 axes of rotation. Stepper motors can hold loads at different steps which is necessary in order for the motors to hold the telescope stationary. Additionally, stepper motors are great when high forces are not needed and the telescope in question weighs less than 1kg which makes this a low force application. Our application also does not require super high speeds and steppers have max torque at lower speeds which makes them suitable.

In order to determine if our motors could maintain the load of the telescope and system in both the azimuth and elevation axises. We performed calculations regarding the motors max radial load and determined if the torque from the telescope would be less than the starting torque specification.

Elevation & Azimuth Motors

Motor radial load max = 6 lbs

Mass (total**) =** [Mass (telescope) = 2.205 lbs] + [mass (telescope attachments) = .795 lbs], [mass (counter-weight) = 1 lbs], [mass (shaft) = 1.896 lbs]

Mass (total, elevation**) =** 5.896 lbs **< max radial load =** 6lbs **-> elevation & azimuth motor will withstand load!**

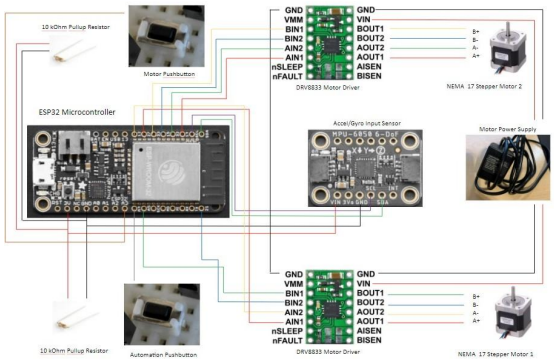
Torque (start**) =** 6.375 in lbs

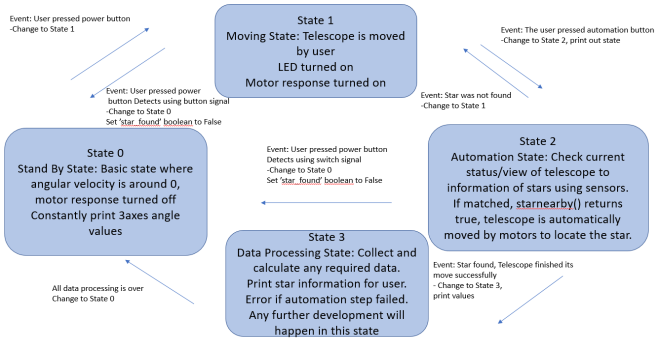
Torque (telescope**) =** 2” * 3lbs **(**weight every component above axis of rotation**) =** 6 in-lbs

Torque (telescope**) < Torque (**starting**) ----> motors will perform!**

One additional consideration for the azimuth motor is balance of moments. The base of the telescope should not rotate together with the azimuth motor. Assuming distance from motor to ground is 60 in, 6.375 lbs * 60” = 383 lbs-in. We need a careful arrangement of bearings and parts to ensure most of the motor’s torque is used for the system above the plate.

Final Circuit Diagram & State Transition Diagram





Reflection

Overall this project was challenging and rewarding.The project checkpoints were constructed well so that our project would progress over the course of the semester. Putting in effort during those checkpoints definitely helped our group make progress. It was also helpful to subdivide the projects and complete them one by one. Regarding what we could have done differently, we could have started with a prototype. This would have given us more chances to test and create an even more robust system.

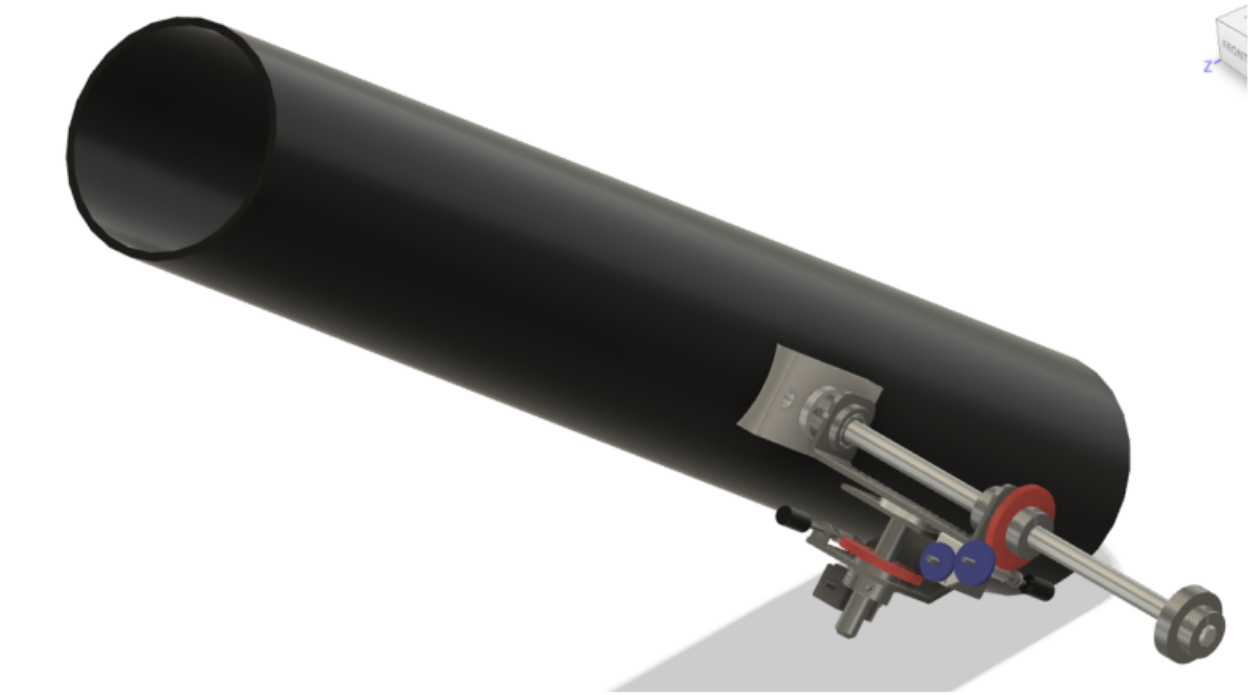
Appendices

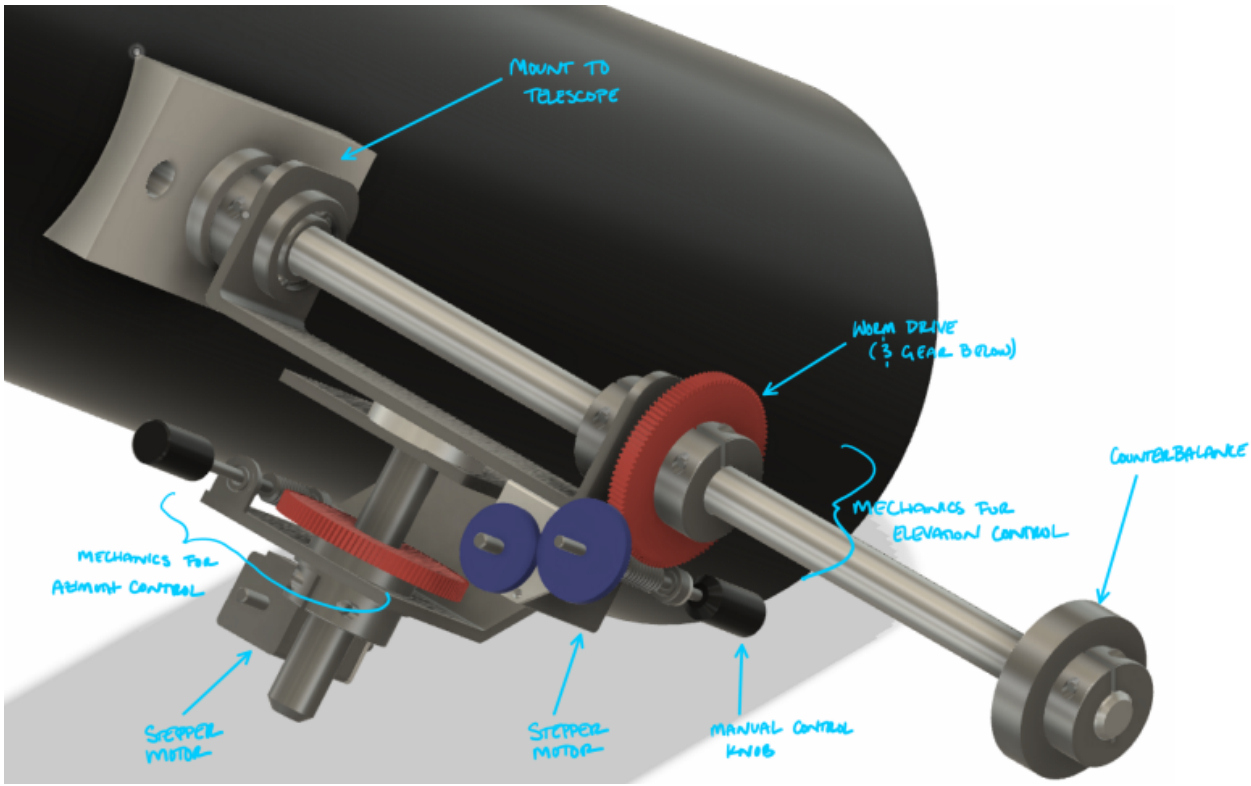
Bill of Materials

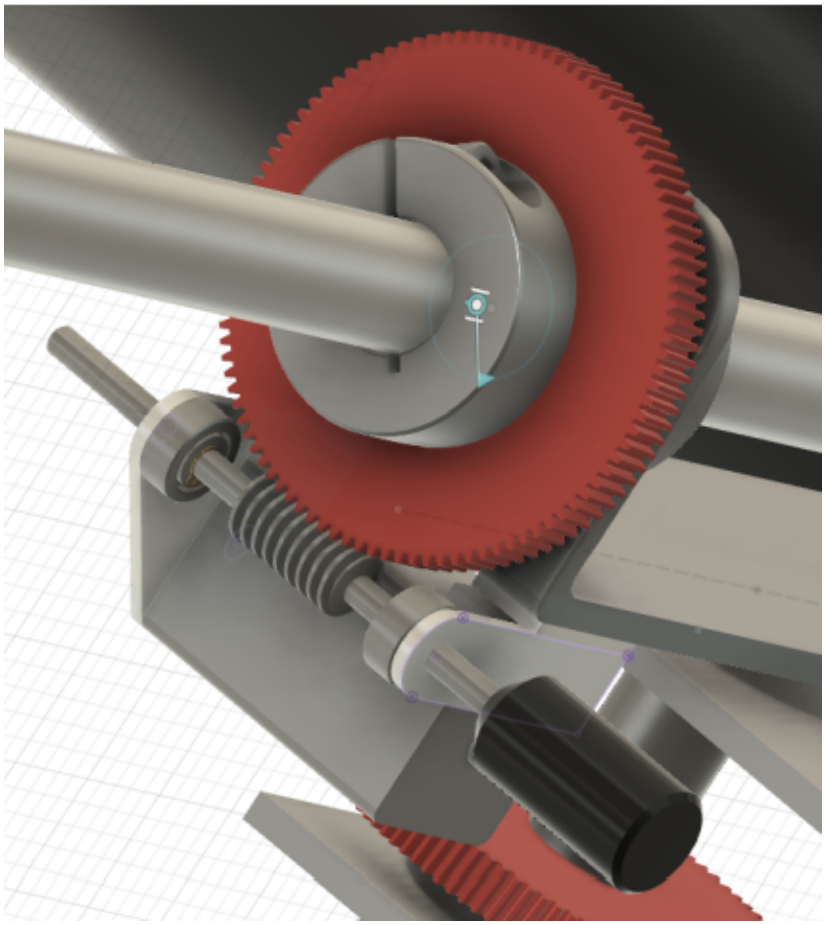
Item Number	Part Name	Quantity	Cost (total	Source Link
#1	NEMA 17 stepper motor (regular size)	1	provided	
#2	NEMA 17 stepper motor (low profile)	1	provided	
#3	DRV8833 Motor Driver	2	provided	
#4	ESP32 Huzzah	1	provided	
#5	Adafruit 6-DOF Gyroscope input sensor	1	$6.95	https://www.adafruit.com/product/3886
#5	Pushbutton	2	provided	
#6	Mini Breadboards	4	$8.81 (tota pack)	Amazon.com: LEGOO 6PCS 170 tie-Points M Breadboard kit : Industrial & Scientific
#7	Aluminum Sheet, .090" x 6" x12"	1	$12.99	No link (Ace Hardware)
#8	K&S ¼" D X 12 in. L Stainless Steel Unthreaded Rod	1	$4.40	K&S 1/4 in. D X 12 in. L Stainles Steel Unthreaded Rod - Ace Hardware
#9	K & S 87147 Round Rod, 1/2 in Dia, 12 in L, Stainless Steel	1	$13.22	K & S 87147 100062362 | Outdo Supply Hardware
#10	K&S 1/2 in. D X 1 ft. L Stainless Steel Tube 1 pk	1	$10.80	K&S 1/2 in. D X 1 ft. L Stainless Steel Tube 1 pk - Ace Hardware
#11	Form Labs standard Resin ($.40 per ml)	27	$10.80	Form Labs standard Resin (per m – Invention Lab Material Store
#12	Markforged Carbon Fiber Material ($0.5/cubic cm	26.41	$13.21	none
#13	24" x 24" x 0.063" 6061 Aluminum Plate (FabLight)	1	$19.20	24" x 24" x 0.063" 6061 Aluminu Plate (FabLight) – Jacobs Hall Material Store

#14	Thrust Ball Bearing for 1/2" Shaft Diameter, 15/16" OD, 0.249" Thick		$5.96	Thrust Ball Bearing, for 1/2" Sha iameter, 15/16" OD, 0.249" Thic McMaster-Carr
#15	Belleville Disc Spring or 1/2" Shaft Diameter, 0.505" ID, 1" OD, 0.0350 Thick		$8.30	elleville Disc Spring, for 1/2" Sha iameter, 0.505" ID, 1" OD, 0.035 Thick | McMaster-Carr
#16	Oil-Embedded 841 Bronze Sleeve Bearing for 3mm Shaft Diameter and 6mm Housing ID, 10mm Long		$6.85	Oil-Embedded 841 Bronze Sleev Bearing, for 3mm Shaft Diamete nd 6mm Housing ID, 10mm Lon McMaster-Carr
#17	Tight-Tolerance 12L14 Carbon Steel Rod Ultra-Machinable, 3 mm Diameter		$6.40	ght-Tolerance 12L14 Carbon Ste Rod, Ultra-Machinable, 3 mm Diameter | McMaster-Carr
#18	Light Duty Dry-Running Nylon Sleeve Bearing Flanged, for 1/4" Shaft Diameter and 3/8" Housin ID, 5/16" Long		$2.76	Light Duty Dry-Running Nylon Sleeve Bearing, Flanged, for 1/4 Shaft Diameter and 3/8" Housin ID, 5/16" Long | McMaster-Carr
#19	1018-1045 Carbon Steel Machine Key Stock 1/8" x 1/8", 12" Long, Oversized		$0.68	1018-1045 Carbon Steel Machin Key Stock, 1/8" x 1/8", 12" Long Oversized | McMaster-Carr
#20	VIGRUE 315Pcs Snap Ring Shop Assortment Alloy Steel External Circli Snap Retaining Clip Rings Set, 15 Sizes		$9.57	VIGRUE 315Pcs Snap Ring Sho Assortment Alloy Steel External Circlip Snap Retaining Clip Ring Set, 15 Sizes: Amazon.com: Industrial & Scientific
#21	Sunluway 10 Pack Flanged Ball Bearing		$15.60	Sunluway 10 Pack Flanged Ball Bearing 1/2" x 1-1/8" x 1/2", Pre Lubricated, Suitable for Lawn Mower, Wheelbarrows, Carts & Hand Trucks Wheel Hub, Replacement for Honda, 12118 Rotary 324 Etc: Amazon.com: Industrial & Scientific
#22	M2-M5 Screw Set	1	$26.99	ttps://www.amazon.com/VIGRU tainless-Washers-Assortment-W ches/dp/B07PJQC7T6/ref=sr_1 ?crid=2CLNNTK8MGZX2&keyw s=metric+screw+assortment&qi 639363351&sprefix=metric+scre +assor%2Caps%2C245&sr=8-2

Images of the CAD, showing mechanical transmission elements (updated from P2/P3) o







Event-Driven Programmed Final Code

```
#include <ESP32Encoder.h>
#include<Wire.h>
#include <Stepper.h>
#define BIN_1 26
#define BIN_2 25
#define LED_PIN 13
#define BTN_STOP 34
#define BTN_AUT 36

// DRV8833 Motor Driver Pins
#define BIN1 4  //BIN1 default LOW, PWM can be applied
#define BIN2 21  //BIN2
#define AIN2 12  //AIN2
#define AIN1 27  //AIN1

#define BIN1A 33 //A0
#define BIN2A 15 //A1
#define AIN2A 32  //A5
#define AIN1A 14 //21

ESP32Encoder encoder;

byte state = 0;
int star = 0;
bool is_star = false;

//constants for sensor
const int MPU_ADDR = 0x68; // I2C address of the MPU-6050
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; //sensor variables declare
int minVal=265;
int maxVal=402;
double x;
double y;
double z;
```

```
//constants for motor
const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolutio
const int steps = 5;
const int stopSteps = 0;

// initialize the stepper library
Stepper myStepper1(stepsPerRevolution, BIN1, BIN2, AIN2, AIN1);
Stepper myStepper2(stepsPerRevolution, BIN1A, BIN2A, AIN2A, AIN1A);

//Setup interrupt variables ----------------------------
volatile int count = 0; // encoder count
volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
volatile bool buttonIsPressed = false;
volatile bool buttonIsPressed_aut = false;

// setting PWM properties ----------------------------
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;

//Initialization -----------------------------------
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  interruptCounter = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
}
```

```
void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder.getCount( );
  encoder.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}

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

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

void setup() {
  // put your setup code here, to run once:
  pinMode(BTN_STOP, INPUT); // configures the specified pin to behave either as an input or an output
  pinMode(LED_PIN, OUTPUT);
  attachInterrupt(BTN_STOP, isr, RISING);
  attachInterrupt(BTN_AUT, isr_aut, RISING);

  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
  encoder.attachHalfQuad(33, 36); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching

  // configure LED PWM functionalitites
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);
```

```
  // initilize timer
  timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns * TIMGn_T
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 2000000, true); // 2000000 * 1 us = 2 s, autoreload true

  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_T
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload true

  //gyro sensor
   Wire.begin(23, 22, 100000); // sda, scl, clock speed
   Wire.beginTransmission(MPU_ADDR);
   Wire.write(0x6B);  // PWR_MGMT_1 register
   Wire.write(0);     // set to zero (wakes up the MPU-6050)
   Wire.endTransmission(true);

   //motor set the speed at 60 rpm:
   myStepper1.setSpeed(10);
   myStepper2.setSpeed(10);

   // at least enable the timer alarms
   timerAlarmEnable(timer0); // enable
   timerAlarmEnable(timer1); // enable

   Serial.println("Automation Telescope System Turned On");


oid loop() {
  //gyro sensor reading
   Wire.beginTransmission(MPU_ADDR);
   Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
   Wire.endTransmission(true);
   Wire.beginTransmission(MPU_ADDR);
   Wire.requestFrom(MPU_ADDR, 14, true); // request a total of 14 registers
   AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
```

```
 AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
 AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
 AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
 int xAng = map(AcX,minVal,maxVal,-90,90);
 int yAng = map(AcY,minVal,maxVal,-90,90);
 int zAng = map(AcZ,minVal,maxVal,-90,90);

 x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI);
 y= RAD_TO_DEG * (atan2(-xAng, -zAng)+PI);
 z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);

 if(state == 0 or state == 1){
 Serial.print("AngleX= ");
 Serial.println(x);

 Serial.print("AngleY= ");
 Serial.println(y);

 Serial.print("AngleZ= ");
 Serial.println(z);
 Serial.println("-----------------------------------------");
 delay(1000);
 }

// switch states
switch (state) {

  case 0 : // motor is stopped, print sensor values
    if (buttonPressEvent()) {
      state = 1;
      startMotorResponse();
      Serial.println("Turn on motor response");

    }
    break;
```

```
case 1 : // motor running
  if (buttonPress_aut()) {
    state = 2;
    Serial.println("Automation State");
    delay(2000);
  }


  if (buttonPressEvent()) {
  state = 0;
  stopMotorResponse();
  Serial.println("Turn off motor response");
  }
  break;

  default: // should not happen
  Serial.println("SM_ERROR");
  break;

case 2 : // autmoation state
  if (starnearby()){ // add service function of motor movements
    state = 3;
    tele_auto();
    star_coordinate();
  }
  else{
    state = 1;
    star_not_found();
  }
  break;

case 3 : // Data Processing: print information of the star found
  if (star_found()) {
    state = 1;
    if(star == 1){ // check for which star you found
```

```
        state = 1;
        if(star == 1){ // check for which star you found
        star_info_moon();
        }
        else if(star == 2){
        star_info_venus();
        }
      }
      else{
        aut_control_error();
      }
      break;
  }
}

//Event Checkers
bool buttonPressEvent() {
  if (buttonIsPressed == true){
    buttonIsPressed = false;
    return true;
  }
  else {
    return false;
  }
}

bool buttonPress_aut() {
  if (buttonIsPressed_aut == true){
    buttonIsPressed_aut = false;
    return true;
  }
  else {
    return false;
  }
}
```

```
bool starnearby() { // compare sensor values to star coordinates
  if ( 170<x && x<190 && 165<y && y<185 && 270<z && z<360 ){  //moon
    return true;
  }
  else if ( 200<x && x<240 && 150<y && y<200 && 220<z && z<330 ){  //venus
    return true;
  }
  else{
    return false;
  }
}


bool star_found() { //star found flag set to true for testing
  if (is_star == true){
    is_star = false;
    return true;
  }
  else {
    return false;
  }
}

//Event Service Responses
void stopMotorResponse() {
  ledcWrite(ledChannel_2, LOW);
  ledcWrite(ledChannel_1, LOW);
  digitalWrite(LED_PIN, LOW);
  delay(2000);
}

void startMotorResponse() {
  ledcWrite(ledChannel_1, LOW);
```

```
void startMotorResponse() {
  ledcWrite(ledChannel_1, LOW);
  ledcWrite(ledChannel_2, 0);
  digitalWrite(LED_PIN, HIGH);
  delay(2000);
}

void tele_auto(){

  if( 160<x && x<200 && 155<y && y<200 && 270<z && z<360 ){    // is
      while(z < 330){
      myStepper2.step(steps);
      read_sensor();
    }
    while(160 < x && x< 175 && 155 < y && y<175){
      myStepper1.step(steps);
      read_sensor();
    }
    while(185<x && x<200 && y>180 && y<200){
      myStepper1.step(-steps);
      read_sensor();
    }

    is_star = true;
  }
  else if ( 190<x && x<250 && 150<y && y<200 && 220<z && z<330 ){

    while(z<250){
      myStepper2.step(steps);
      read_sensor();
    }
    while(z>300){
      myStepper2.step(-steps);
      read_sensor();
    }
```

```
      read_sensor();
    }
    while(190 < x && x< 210 && 150 < y && y<170){
      myStepper1.step(steps);
      read_sensor();
    }
    while(230<x && x<250 && y>180 && y<200){
      myStepper1.step(-steps);
      read_sensor();
    }
    is_star = true;
  }
  else{
    is_star = false;
  }

}

void star_coordinate(){
    if ( 170<x && x<190 && 165<y && y<185 && 330<z && z<360 ){  //moon
    star = 1;
  }
  else if ( 200<x && x<240 && 150<y && y<200 && 250<z && z<300 ){  //venus
    star = 2;
  }
}

void star_info_moon(){ //star 1 is moon
  Serial.println("        Name: Moon");
  Serial.println("Distance From Earth: 384472 km");
  Serial.println("Surface Temperature: -153°C ~ 107°C");
  Serial.println("Interesting Facts: Moon is Earth's only natural satellite");
  Serial.println("On 2020, water was found on Moon!");
  star = 0;
}
```

```
void star_info_venus(){
  Serial.println("        Name: Venus");
  Serial.println("Distance From Earth: 61 million km");
  Serial.println("Surface Temperature: 475°C");
  Serial.println("Interesting Facts: Venus is the second brightest natural object in the night sky");
  Serial.println("A day on Venus is longer than a year!");
  star = 0;
}

void read_sensor(){ // convert acceleration sensor's reading values
   Wire.beginTransmission(MPU_ADDR);
   Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
   Wire.endTransmission(true);
   Wire.beginTransmission(MPU_ADDR);
   Wire.requestFrom(MPU_ADDR, 14, true); // request a total of 14 registers
   AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
   AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
   AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
   int xAng = map(AcX,minVal,maxVal,-90,90);
   int yAng = map(AcY,minVal,maxVal,-90,90);
   int zAng = map(AcZ,minVal,maxVal,-90,90);

   x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI);
   y= RAD_TO_DEG * (atan2(-xAng, -zAng)+PI);
   z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);

   if(state == 0 or state == 1){
   Serial.print("AngleX= ");
   Serial.println(x);

   Serial.print("AngleY= ");
   Serial.println(y);

   Serial.print("AngleZ= ");
   Serial.println(z);
```

```
   Serial.print("AngleZ= ");
   Serial.println(z);
   Serial.println("-----------------------------------------");
   delay(1000);
   }
}

void star_not_found(){
  Serial.println("Nothing found!");
}

void aut_control_error(){
  Serial.println("Automation finding failed!");
}
```