```
// Tracker Calculation
//    Gear Ratio
//      Planetary gear ratio - 4.4:1
//      Planetary stages - 3
//      Drive gear ratio - 28/18:1 or 1.555:1
//      Total Effective Gear Ratio - 132.5084444:1
//    Stepper Delay
//      Number of steps = 4076 * 132.51 = 540104.4
//      Degrees per step = 0.000667
//      Degrees per min = 0.25068
//      Steps per minute = degrees/min / degrees/step = 376.0927
//      Steps per second = 376.0927/60 = 6.268212
//      Delay per step = 0.159535
//      Delay per 2 steps = 0.31907
//    NOTES:
//      May want to consider letting the stepper fire one move per delay
//      May want to calculate the period based on steps and multiplier


#include <ezButton.h>                     // For non-blocking switch debounce
#include <CheapStepper.h>                 // For non-blocking stepper motor control

// Object Declarations

CheapStepper stepper (8,9,10,11);         // Stepper motor object
ezButton trackToggle(5);                  // Toggle switch to turn tracking on and off
ezButton slewClockwiseMoment(4);          // Moment switch to slew the camera up
ezButton slewCounterMoment(3);            // Moment switch to slew the camera down

// Global Variable Declarations

bool slewClockwise = true;        // Initialize the slew direction
bool trackInit = false;           // State variable for initialization status; switch changes state to on
bool slewClockwiseInit = false;   // State variable for initialization status; switch changes state to on
bool slewCounterInit = false;     // State variable for initialization status; switch changes state to on
bool resetFlag = false;           // Flag to indicate that a reset is needed; switch changes status to off
bool blueState = LOW;             // State variable to drive LED on/off
const long trackingPeriod = 319;  // Tracker movement wait period
const long slewPeriod = 5;        // Slew movement wait period
int slewSteps = 2;                // Number of steps to slew stepper motor
int blueLED = 7;                  // Blue LED is wired to digital pin 7
int redLED = 6;                   // Red LED is wired to digital pin 6
unsigned long trackingTimer = 0;  // Store previous tracking period start
unsigned long slewTimer = 0;      // Store previous slew period start
```

```
void setup() {
  trackToggle.setDebounceTime(50);             // Set the toggle debounce time to 50ms
  slewClockwiseMoment.setDebounceTime(50);     // Set the moment debounce time to 50ms
  slewCounterMoment.setDebounceTime(50);       // Set the moment debounce time to 50ms

  pinMode(blueLED, OUTPUT);             // Blue LED pin is an output - draws 3.3mA at 3.3V
  pinMode(redLED, OUTPUT);              // Red LED pin is an output - draws 3.3mA at 3.3V
  digitalWrite(redLED, HIGH);           // Show device is powered on by turning red LED on

  stepper.set4076StepMode(); // Set stepper motor steps per revolution to 4076 (measured) instead 4096
  stepper.setRpm(16);                   // Set the stepper RPM to 16 (default)
}

void loop() {
  // Load Switch Information

  trackToggle.loop();                   // Check the status of the tracking toggle switch
  slewClockwiseMoment.loop();           // Check the status of the clockwise slewing moment switch
  slewCounterMoment.loop();             // Check the status of the clockwise slewing moment switch

  // Non-blocking stepper motor requirement

  stepper.run();

  // Define Local Variables

  int trackToggleState = trackToggle.getState();                  // 1 is off, 0 is on (flipped)
  int slewClockwiseMomentState = slewClockwiseMoment.getState();  // 1 is off, 0 is on (pressed)
  int slewCounterMomentState = slewCounterMoment.getState();      // 1 is off, 0 is on (pressed)
  unsigned long currentTimer = millis();  // Hold the current hardware timer ms count since program start

  // Loop() Code to Execute
  // If else loops ensure only 1 button can be active at a time in order of priority

  if (trackToggleState == 0) {   // If the track toggle is ON
    if (trackInit == false) {    // If this the is first tracking cycle, initialize the tracking state
      SwitchInit(trackInit, true, true);  // Initialize the tracker
    }

    if (currentTimer - trackingTimer >= trackingPeriod) {     // Check if it's time to move
      trackingTimer = currentTimer;                           // Reset the current timer
      stepper.newMove(slewClockwise, slewSteps);  // Issue the non-blocking move command to the stepper
    } else if (currentTimer - trackingTimer >= trackingPeriod/3) {// If 1/3 of the wait period has elapsed
      stepper.off(); // Don't leave motor in high current state for whole wait - no chance of backdrive
    }
  } else if (slewClockwiseMomentState == 0) {   // If the slew clockwise button is pressed
    if (slewClockwiseInit == false) { // If this the is first slew up cycle, initialize the tracking state
      SwitchInit(slewClockwiseInit, true, false);      // Initialize the slew up initialization
    }

    if (currentTimer - slewTimer >= slewPeriod) { // Check if it's time to move
      slewTimer = currentTimer;                   // Reset the slew timer
      blueState = !blueState;                     // Change blue LED State - blue LED will strobe
      digitalWrite(blueLED, blueState);           // Write new blue LED State
      stepper.newMove(slewClockwise, slewSteps);  // Issue the non-blocking move command to the stepper
    }
  } else if (slewCounterMomentState == 0) {       // If the slew counter-clockwise button is pressed
    if (slewCounterInit == false) {  // If this the is first slew up cycle, initialize the tracking state
      SwitchInit(slewCounterInit, false, false);  // Initialize the slew up initialization
    }

    if (currentTimer - slewTimer >= slewPeriod) {  // Check if it's time to move
      slewTimer = currentTimer;                    // Reset the slew timer
      blueState = !blueState;                      // Change blue LED State - blue LED will strobe
      digitalWrite(blueLED, blueState);            // Write new blue LED State
      stepper.newMove(slewClockwise, slewSteps);   // Issue the non-blocking move command to the stepper
    }
  } else {
    // No buttons are pushed - reset initialization states if the reset flag is true
    if (resetFlag == true) {
      SwitchReset();
    }
  }
}

void SwitchInit (bool &componentInit, bool slewDirection, bool setLED) {
  componentInit = true;
  slewClockwise = slewDirection;

  // Set reset state variable here
  resetFlag = true;

  if (setLED == true) {
    // Blue LED should stay on when tracker is tracking
    blueState = setLED;
    digitalWrite(blueLED, blueState);
  }
}

void SwitchReset () {
  trackInit = false;
  slewClockwiseInit = false;
  slewCounterInit = false;

  stepper.stop();
  stepper.off();
  resetFlag = false;

  blueState = LOW;                    // Set the blue LED state to off
  digitalWrite(blueLED, blueState);   // Write the state to the output LED
}
```