```
#include <Mouse.h>

const int buttonPinRight = 2;  // Right click button pin (Connect to digital pin 2)
const int buttonPinLeft = 3;   // Left click button pin (Connect to digital pin 3)
const int buttonPinDrag = 4;   // Drag button pin (Connect to digital pin 4)

int buttonStateLeft = HIGH;
int lastButtonStateLeft = HIGH;
int buttonStateRight = HIGH;
int lastButtonStateRight = HIGH;
int buttonStateDrag = HIGH;
int lastButtonStateDrag = HIGH;

unsigned long lastDebounceTimeLeft = 0;
unsigned long lastDebounceTimeRight = 0;
unsigned long lastDebounceTimeDrag = 0;
unsigned long debounceDelay = 50;

bool isDragging = false;  // Flag to track if the drag action is active

const int buttonUpPin = 5;    // Connect the up button to digital pin 5
const int buttonLeftPin = 6;  // Connect the left button to digital pin 6
const int buttonRightPin = 7; // Connect the right button to digital pin 7
const int buttonDownPin = 8;  // Connect the down button to digital pin 8

int mouseSpeed = 5; // Adjust the speed of mouse movement

void setup() {

  pinMode(buttonPinLeft, INPUT);
  pinMode(buttonPinRight, INPUT);
  pinMode(buttonPinDrag, INPUT);

  digitalWrite(buttonPinLeft, HIGH);  // Enable internal pull-up resistor
  digitalWrite(buttonPinRight, HIGH); // Enable internal pull-up resistor
  digitalWrite(buttonPinDrag, HIGH);  // Enable internal pull-up resistor

  Mouse.begin();

  pinMode(buttonLeftPin, INPUT_PULLUP);  // Set pin 5 as an input with internal pull-up resistor
  pinMode(buttonRightPin, INPUT_PULLUP); // Set pin 6 as an input with internal pull-up resistor
```

```
  pinMode(buttonUpPin, INPUT_PULLUP);    // Set pin 7 as an input with internal
pull-up resistor
  pinMode(buttonDownPin, INPUT_PULLUP);  // Set pin 8 as an input with internal
pull-up resistor

  Serial.begin(9600);
}

void loop() {

  int readingLeft = digitalRead(buttonPinLeft);
  int readingRight = digitalRead(buttonPinRight);
  int readingDrag = digitalRead(buttonPinDrag);

  // Left button handling
  if (readingLeft != lastButtonStateLeft) {
    lastDebounceTimeLeft = millis();
  }

  if ((millis() - lastDebounceTimeLeft) > debounceDelay) {
    if (readingLeft != buttonStateLeft) {
      buttonStateLeft = readingLeft;

      if (buttonStateLeft == HIGH) {
        Mouse.click();
      }
    }
  }

  lastButtonStateLeft = readingLeft;

  // Right button handling
  if (readingRight != lastButtonStateRight) {
    lastDebounceTimeRight = millis();
  }

  if ((millis() - lastDebounceTimeRight) > debounceDelay) {
    if (readingRight != buttonStateRight) {
      buttonStateRight = readingRight;

      if (buttonStateRight == HIGH) {
        Mouse.click(MOUSE_RIGHT);
      }
    }
  }
```

```
lastButtonStateRight = readingRight;

// Drag button handling
if (readingDrag != lastButtonStateDrag) {
  lastDebounceTimeDrag = millis();
}

if ((millis() - lastDebounceTimeDrag) > debounceDelay) {
  if (readingDrag != buttonStateDrag) {
    buttonStateDrag = readingDrag;

    if (buttonStateDrag == HIGH) {
      if (isDragging) {
        Mouse.release(MOUSE_LEFT);  // Release left button to stop dragging
        isDragging = false;
      } else {
        isDragging = true;
        Mouse.press(MOUSE_LEFT);  // Press left button to start dragging
      }
    }
  }
}

lastButtonStateDrag = readingDrag;

int buttonLeftState = digitalRead(buttonLeftPin);
int buttonRightState = digitalRead(buttonRightPin);
int buttonUpState = digitalRead(buttonUpPin);
int buttonDownState = digitalRead(buttonDownPin);

if (buttonLeftState == HIGH) {
  Mouse.move(-mouseSpeed, 0);
}

if (buttonRightState == HIGH) {
  Mouse.move(mouseSpeed, 0);
}

if (buttonUpState == HIGH) {
  Mouse.move(0, -mouseSpeed);  // Move the cursor up
}

if (buttonDownState == HIGH) {
  Mouse.move(0, mouseSpeed);   // Move the cursor down
```

```
  }

  delay(10); // Adjust this value for the desired update rate
}
```