```cpp

/\*===================================VERSION3=========================================

A red ball is glued on top of the joystick.

The joystick is north of the rotary encoder.

The pins of the joystick are facing south.

The schematic needs to be updated\*/

#include <Mouse.h>

#include <Encoder.h>

#include <ezLED.h>

//-------------------------pinout variables

const int VRx = A1; //joystick xAxis

const int VRy = A0; //joystick yAxis

#define joystickVCC 21 //joystick voltage source

#define SW 7 //joystick push button

#define LEDAmber 4 //Amber LED

#define LEDWhite 5 //White LED

#define LEDRed 8 //Red LED

#define LEDBlue 9 //Purple LED

#define BTNBlack 6 //black momentary button

Encoder knobLeft(2,3);

//---------------------sets global variables

int analogVRx; //reading from the joysticks horizontal axis

int analogVRy; //reading from the joysticks vertical axis

int xAxis; //horizontal movement range

int yAxis; //accelerates horizontal joystick movement

int yAccel; //vertical movement range

int xAccel; //accelerates vertical joystick movement

int brightness; //LED PWM brightnesslevel

int incrementSteps; //encoder output increments (Higher = Faster)

int tickInterval; //[ms] between encoder clicks

//---------------------these defaults can only be changed here

const int coarse = 3; //increment steps during a faster rotation

const int fine = 0; //increment steps during a slower rotation

const int normal = 1; //increment steps during a normal rotation

const int accelHigh = 1016; //positive acceleration threshold

const int accelLow = 0; //negative acceleration threshold

const int rotaryThreshLOW = 1; //[ms] between rotary steps to be considered fast

const int rotaryThreshHIGH = 20; //[ms] between rotary steps to be considered slow

const int joystickAccelMAX = 2; //speed of the cursor when the joystick is at limit

const int thresholdHigh = 540; //sets a minimum threshold to convert the analog reading into directional input

const int thresholdLow = 500; //sets a minimum threshold to convert the analog reading into directional input

const int latchTimeOut = 400; //interval before the left mouse unlatches after rotation

const int tickHold = 1; //[ms] hold for the red LED

const int delayResponse = 5; //sets a small delay at the end of the loop for good measure

const int fadeAmount = 3; //how many values of fade increment

const int fadeRate = 30; //how quickly between fade out increments

//------------------sets global status indicators

bool tick; //marks each rotary click

bool latch; //left click latch

bool cursorRight; //issues move right command

bool cursorLeft; //issues move left command

bool cursorUp; //issues move up command

bool cursorDown; //issues move down command

bool blackButtonStatus; //LOW = pressed, HIGH = open

bool SWStatus; //LOW = pressed, HIGH = open

bool axisStatus; //LOW = horizontal, HIGH = vertical

long rotaryPosition;

//the following are a part of millis() to help control some timing variables

unsigned long fadeMillis; //current time in milliseconds

unsigned long lastFadeMillis; //last time it was called

unsigned long lastButtonPress = 0; //resets the counter the last time the SW was pressed.

unsigned long rotationTime; //this variable is used to record the time between rotation ticks

unsigned long rotationInterval; //microseconds elapsed between ticks

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//---------------------------------PROGRAM FUNCTIONS-----------------------

void setup() {

attachInterrupt(digitalPinToInterrupt(SW), ISRAxisToggle, FALLING);

initialize();

pinMode();

Serial.begin(9600);

Mouse.begin();}

void pinMode(){

pinMode(BTNBlack, INPUT\_PULLUP);

pinMode(SW, INPUT\_PULLUP);

pinMode(joystickVCC, OUTPUT);

pinMode(LEDAmber, OUTPUT);

pinMode(LEDWhite, OUTPUT);

pinMode(LEDRed, OUTPUT);

pinMode(LEDBlue, OUTPUT);}

void initialize(){

digitalWrite(joystickVCC, HIGH); //provides voltage for the joystick

int LEDAmberState = LOW;

int analogVRy = analogRead(VRx); //initializes a reading from the joystick's vertical movement;

int analogVRx = analogRead(VRy); //initializes a reading from the joystick's horizontal movement;

int brightness = 0; // how bright the LED is

int fadeAmount = 5; // how many points to fade the LED by

fadeMillis = millis();

lastFadeMillis = 0;

cursorUp = 0;

cursorDown = 0;

cursorLeft = 0;

cursorRight = 0;

xAxis = 0;

yAxis = 1;

axisStatus = 1;

xAccel = 0;

yAccel = 0;

rotationTime = millis();

rotationInterval = micros();

latch = 0;

tick = 0;

brightness = 255;

initLightShow();}

void loop(){

digitalFeedback(); //debug reads and prints digitalPin status

analogFeedback(); //debug reads and prints digitalPin status

booleanFeedback(); //debug displays, LEFT, RIGHT, UP, DOWN as TRUE/FALSE

axisFeedback(); //debug displays axis status; LOW = HOR, HIGH = VER

accelFeedback(); //debug joystick acceleration

rotaryAccelFeedback(); //debug rotary accel

VRxResponse(); //decides cursor movement based off of the VRx reading

VRyResponse(); //decides cursor movement based off of the VRy reading

cursorAccel(); //decides cursor acceleration.

joystickCommands(); //checks variables and commands the mouse

rotaryCommands(); //checks variables and commands vertical movement

rotationAcceleration(); //checks rotation speed and adds increments

blackButtonCommands(); //checks black button status

leftClick(); //controls the command to the left mouse click

axisToggleCommands(); //decides and executes rotary axis toggle

digitalWrite(LEDAmber, LOW); //resets the LEDAmber

latchRelease(); //releases the left click latch after set time

leftClickLEDAmber(); //LEDAmber responds to left click

redFlash(); //LEDRed responds to joystick movement

blueClick(); //LEDBlue reponds to ticks

delay(delayResponse); //small delay at the end of the loop

}

//---------------------------------SERIAL FEEDBACK-------------------------

void digitalFeedback(){

Serial.println("dVRx");

Serial.println(digitalRead(VRx));

Serial.println("dVRy");

Serial.println(digitalRead(VRy));

Serial.println("dSW");

Serial.println(digitalRead(SW));

}

void analogFeedback(){

Serial.println("aVRx");

Serial.println(analogRead(VRx));

Serial.println("aVRy");

Serial.println(analogRead(VRy));

Serial.println("aSW");

Serial.println(analogRead(SW));

Serial.print(xAccel);

Serial.println(yAccel);

}

void booleanFeedback(){

Serial.print(cursorLeft);

Serial.print(cursorRight);

Serial.print(cursorDown);

Serial.print(cursorUp);

}

void axisFeedback(){

Serial.print(xAxis);

Serial.println(yAxis);

}

void accelFeedback(){

Serial.print(xAccel);

Serial.println(yAccel);

}

void rotaryAccelFeedback(){

Serial.print(incrementSteps);

Serial.print(" - ");

Serial.println(tickInterval);

}

void tickIntervalFeedback(){

unsigned long interval = micros() - rotationInterval;

if (interval > 2000)

Serial.println(interval);

rotationInterval = micros();

}

//----------------------------------EVENT FUNCTIONS------------------------

void VRxResponse(){

int analogVRx = analogRead(VRx); //samples the analog voltage from the VRx pin (orientation is configurable)

if (analogVRx > thresholdHigh){ //default cursor RIGHT

cursorRight = 1;

cursorLeft = 0;

}

else if(analogVRx < thresholdLow){ //default cursor LEFT

cursorRight = 0;

cursorLeft = 1;

}

else {

cursorRight = 0;

cursorLeft = 0;

}

}

void VRyResponse(){

int analogVRy = analogRead(VRy); //samples the analog voltage from the VRy pin (orientation is configurable)

if (analogVRy > thresholdHigh){ //default cursor DOWN

cursorDown = 1;

cursorUp = 0;

}

else if(analogVRy < thresholdLow){ //default cursor UP

cursorDown = 0;

cursorUp = 1;

}

else {

cursorDown = 0;

cursorUp = 0;

}}

void cursorAccel(){

//------the following adds acceleration ----------

if (analogRead(VRx) >= accelHigh){

xAccel = joystickAccelMAX;

}

else if (analogRead(VRx) == accelLow){

xAccel = joystickAccelMAX;

}

else{

xAccel = 0;

}

if (analogRead(VRy) >= accelHigh){

yAccel = joystickAccelMAX;

}

else if (

analogRead(VRy) == accelLow){

yAccel = joystickAccelMAX;

}

else{

yAccel = 0;

}}

void joystickCommands(){

if (cursorRight == 1){

Mouse.move(1+xAccel,0,0);

}

if (cursorLeft == 1){

Mouse.move(-(1+xAccel),0,0);

}

if (cursorDown == 1){

Mouse.move(0,-(1+yAccel),0);

}

if (cursorUp == 1){

Mouse.move(0,1+yAccel,0);

}

}

void rotaryCommands(){

long newRotaryPosition;

newRotaryPosition = knobLeft.read();

if (newRotaryPosition != rotaryPosition) {

latch = 1;

rotationTime = millis();

rotationAcceleration(); //check the rate of encoder spin to determine cursor speed

if(newRotaryPosition > rotaryPosition){

Mouse.move(-xAxis,yAxis,0);

}

else if(newRotaryPosition < rotaryPosition){

Mouse.move(xAxis,-yAxis,0);

}

rotaryPosition = newRotaryPosition;

}

}

void blackButtonCommands(){

if (digitalRead(BTNBlack) == LOW){

blackButtonStatus = 0;

}

else{

blackButtonStatus = 1;

}}

void leftClick(){

if (!blackButtonStatus || latch == HIGH){

Mouse.press(MOUSE\_LEFT);

}

else{

Mouse.release(MOUSE\_LEFT);

}

}

void ISRAxisToggle(){

if (millis() - lastButtonPress >= 200) {

axisStatus = !axisStatus; //toggles the rotary axis between horizontal and vertical

lastButtonPress = millis();

}

}

void axisToggleCommands(){

if (axisStatus == 0){

xAxis = 0;

yAxis = 1+incrementSteps;

digitalWrite(LEDWhite, LOW);

}

else {

xAxis = 1+incrementSteps;

yAxis = 0;

digitalWrite(LEDWhite, HIGH);

}

}

void rotationAcceleration(){

tickInterval = (millis() - rotationTime);

if (tickInterval < rotaryThreshLOW){

incrementSteps = coarse;}

else if (tickInterval > rotaryThreshHIGH){

incrementSteps = fine;}

else {incrementSteps = 0;}

}

void latchRelease(){

if (millis() - rotationTime > latchTimeOut){

latch = LOW;

}

}

//----------------------------------LED FUNCTIONS------------------------------

void fiveBlinksAnalog(){

int i = 0;

int msHold = 30;

while(i<5){ //flashes a light 5 times

analogWrite(LEDWhite, 125);

analogWrite(LEDAmber, 0);

analogWrite(LEDBlue, 0);

analogWrite(LEDRed, 0);

delay(msHold);

analogWrite(LEDAmber, 255);

delay(msHold);

analogWrite(LEDBlue, 255);

delay(msHold);

analogWrite(LEDRed, 255);

delay(msHold);

analogWrite(LEDRed, 0);

delay(msHold);

analogWrite(LEDBlue, 0);

delay(msHold);

analogWrite(LEDAmber, 0);

delay(msHold);

analogWrite(LEDWhite, 0);

delay(msHold);

i++;

}}

void fiveBlinksDigital(){

int i = 0;

while(i<5){ //flashes a light 5 times

digitalWrite(LEDAmber, HIGH);

digitalWrite(LEDWhite, LOW);

digitalWrite(LEDRed, HIGH);

digitalWrite(LEDBlue, LOW);

delay(100);

digitalWrite(LEDAmber, LOW);

digitalWrite(LEDWhite, HIGH);

digitalWrite(LEDRed, LOW);

digitalWrite(LEDBlue, HIGH);

delay(100);

digitalWrite(LEDRed, HIGH);

digitalWrite(LEDWhite, HIGH);

digitalWrite(LEDAmber, HIGH);

delay(200);

digitalWrite(LEDBlue, LOW);

delay(200);

digitalWrite(LEDRed, LOW);

delay(200);

digitalWrite(LEDWhite, LOW);

delay(200);

digitalWrite(LEDAmber, LOW);

i++;

}}

void initLightShow(){

//fiveBlinksDigital(); //5 quick blinks showing the initialize process

fiveBlinksAnalog();

analogWrite(LEDWhite, 255); //a single long flash to show initialize finish

analogWrite(LEDAmber, 255); //a single long flash to show initialize finish

analogWrite(LEDBlue, 255); //a single long flash to show initialize finish

analogWrite(LEDRed, 255); //a single long flash to show initialize finish

delay(1000);

analogWrite(LEDAmber, 0); //remember to turn the lights off?

}

void leftClickLEDAmber(){

digitalWrite(LEDAmber, Mouse.isPressed(MOUSE\_LEFT));}

void redFlash(){

if (cursorDown || cursorUp || cursorLeft || cursorRight == 1){

digitalWrite(LEDRed, HIGH);

}

else{

digitalWrite(LEDRed, LOW);

}}

void blueClick(){

if (millis() - rotationTime < tickHold){

analogWrite(LEDBlue, 10);

//tickIntervalFeedback();

}

else{

analogWrite(LEDBlue, 0);

}

}

```