/\*

v0.81
Claw machine game with logic to accept coin, wait for player start and run game
gantry can only take one input at a time. (IE no diagonal movement is possible).

(c) 2015 Ryan Bates, RetroBuiltGames.com
Retrobuiltgames.com

\*/
const int buttonForward = 31;
const int buttonBackward = 33;
const int buttonLeft = 35;
const int buttonRight = 37;
const int buttonUp = 39;
const int buttonDown = 41;
const int buttonStepperRPMspeed = 43;
const int buttonCoin = 45;
const int buttonStart = 47;
const int servoPower = 8;
const int ledPin = 12;
const int ledPinCoin = 11;

//const int x=0; //x cursor position
//const int y=0; // y cursor position

int StepperRPM = 60; // default speed and hold variable
int StepperRPM\_FAST = 90;
int StepperRPM\_SLOW = 60;

#include <Stepper.h>
#include<Time.h>

int ledStateStart = LOW;
int ledStateCoin = LOW;

int buttonStateForward = HIGH;
int buttonStateBackward = HIGH;
int buttonStateLeft = HIGH;
int buttonStateRight = HIGH;
int buttonStateUp = HIGH;
int buttonStateDown = HIGH;
int buttonStateStepperRPMspeed = HIGH;
int buttonStateCoin = HIGH;
int buttonStateStart = HIGH;

long intervalStart = 500; //rate of blink for 'insert coin' / 'game over'
long intervalIdle = 2000; // rate of blink for 'push start to begin'
long previousMillis = 0;
int y=0;
int x=0;

const int stepsPerRevolution = 200;
Stepper Ystepper(stepsPerRevolution, 22,24,26,28);
Stepper Xstepper(stepsPerRevolution, 30,32,34,36);
Stepper Zstepper(stepsPerRevolution, 38,40,42,44);

#include <Servo.h>
Servo myservo; // create servo object to control a servo
int potpin = 0; // analog pin used to connect the potentiometer
int val;

unsigned int timelimit = 5; //amount of time player can play claw machine

#include<LiquidCrystal.h>
LiquidCrystal lcd (7,6,5,4,3,2);

void setup() {
 setTime(01,01,00,9,9,1999); //(HH,MM,SS,Day, Month, Year)
lcd.begin(20,4);

 pinMode(buttonForward, INPUT);
 pinMode(buttonBackward, INPUT);
 pinMode(buttonLeft, INPUT);
 pinMode(buttonRight, INPUT);
 pinMode(buttonUp, INPUT);
 pinMode(buttonDown, INPUT);
 pinMode(buttonStepperRPMspeed, INPUT);
 pinMode(buttonCoin, INPUT);
 pinMode(buttonStart, INPUT);
 pinMode(ledPin, OUTPUT);
 pinMode(ledPinCoin, OUTPUT);

 pinMode(servoPower, OUTPUT);

 //enable intermal pull ups. Note all functioning logic must trigger low.
 digitalWrite(buttonForward, INPUT\_PULLUP);
 digitalWrite(buttonBackward, INPUT\_PULLUP);
 digitalWrite(buttonLeft, INPUT\_PULLUP);
 digitalWrite(buttonRight, INPUT\_PULLUP);
 digitalWrite(buttonUp, INPUT\_PULLUP);
 digitalWrite(buttonDown, INPUT\_PULLUP);
 digitalWrite(buttonStepperRPMspeed, INPUT\_PULLUP);
 digitalWrite(buttonCoin, INPUT\_PULLUP);
 digitalWrite(buttonStart, INPUT\_PULLUP);
 digitalWrite(servoPower, LOW); //default servo power to be OFF
 digitalWrite(ledPin, LOW);
 digitalWrite(ledPinCoin, LOW);

 myservo.attach(9); //

}

void loop() {

 //-----------Game Idle -------------------

 buttonStateCoin = digitalRead(buttonCoin);
 unsigned long currentMillis = millis(); // check to see if it's time to refresh screen

 if(currentMillis - previousMillis > intervalIdle) {
 previousMillis = currentMillis; // save the last time of cycle execution
 if (y == 0) {
 y= y+1; x=x+1; // add to the y cursor position
 lcd.clear();
 lcd.setCursor(5,y);
 lcd.print("GAME OVER");
 lcd.setCursor(0,y+1);
 lcd.print("Insert $0.25 to Play");
 lcd.setCursor(0,y+1); }

 else{
 lcd.clear();
 lcd.setCursor(5,y+1);
 lcd.print("GAME OVER");
 lcd.setCursor(0,y);
 lcd.print("Insert $0.25 to Play");
 lcd.setCursor(0,3);
 y = y-1; x=x-1; } }

if ( second()== timelimit ) {
 lcd.clear();
 lcd.setCursor(2,1);
 lcd.print ("Time Expired :(");
 myservo.write(0); delay(500); //release claw
 digitalWrite(servoPower, LOW); // Turn OFF servo power

 }

//-----------Coin Accepted-------------------

 if (buttonStateCoin == LOW) //if a quarter is inserted
 { lcd.clear(); // wipe screen
 do {unsigned long currentMillis = millis();

 lcd.setCursor(0,0);
 lcd.print("Credit: $0.25");
 lcd.setCursor(0,1);
 lcd.print("Push Start to Begin");
 lcd.setCursor(0,2);
 lcd.print("Good luck!"); //}

 if(currentMillis - previousMillis > intervalStart) {
 previousMillis = currentMillis; // save the last time you blinked the LED
 if (ledStateStart == LOW) // if the LED is off turn it on and vice-versa:
 ledStateStart = HIGH;
 else
 ledStateStart = LOW;
 digitalWrite(ledPin, ledStateStart); }

 //----wait for "START" -------------------
 buttonStateStart = digitalRead(buttonStart); //wait for player to push start
 } while (buttonStateStart == HIGH); // while start button is not pressed
 runGame(); //if start is pushed exit while loop and full game starts
 }

}
//-------------start of subroutine sections-------------
 void runGame(){

 digitalWrite(ledPin, LOW); //turn blinking Start button LED off
 setTime(01,01,00,9,9,1999); // reset second to valid game play
 lcd.clear(); // wipe screen
 digitalWrite(servoPower, HIGH); //enable servo Power

 //-------------------start of gameplay ------------------------------
 while (second() <timelimit && year() ==1999) {

 lcd.setCursor(3,0);
 lcd.print("Time Remaining:");
 lcd.setCursor(8,2);
 lcd.print(second());
 lcd.setCursor(10,2);
 lcd.print("/"); lcd.print(timelimit);

 buttonStateForward = digitalRead(buttonForward);
 buttonStateBackward = digitalRead(buttonBackward);
 buttonStateLeft = digitalRead(buttonLeft);
 buttonStateRight = digitalRead(buttonRight);
 buttonStateUp = digitalRead(buttonUp);
 buttonStateDown = digitalRead(buttonDown);
 buttonStateStepperRPMspeed = digitalRead(buttonStepperRPMspeed);

 //----------Stepper Speed toggle-----------------
 Ystepper.setSpeed(StepperRPM);
 Xstepper.setSpeed(StepperRPM);
 Zstepper.setSpeed(100);

 if (buttonStateStepperRPMspeed == LOW) {
 StepperRPM = StepperRPM\_FAST; }
 else { StepperRPM = StepperRPM\_SLOW; }

 //------------------X Axis-------------------------------
 while (digitalRead(buttonLeft) == LOW &&
 digitalRead(buttonForward)== HIGH && digitalRead(buttonBackward)== HIGH ){
 Xstepper.step(25);
 lcd.setCursor(8,2);
 lcd.print(second());
 if (second() >= timelimit)
 break; } // end the game without this player can hold directon and
 //the loop will skip over the timelimit check

 while (digitalRead(buttonRight) == LOW &&
 digitalRead(buttonForward)== HIGH && digitalRead(buttonBackward)== HIGH ) {
 Xstepper.step(-25);
 lcd.setCursor(8,2);
 lcd.print(second());
 if (second() >= timelimit)
 break; }

 digitalWrite(38,LOW); //disables hold (no current supplied to stepper when idle)
 digitalWrite(40,LOW);
 digitalWrite(42,LOW);
 digitalWrite(44,LOW);

 //-------Y Axis------------------------------------------
 while (digitalRead(buttonForward) == LOW
 && digitalRead(buttonRight) == HIGH && digitalRead(buttonLeft) == HIGH) {
 Ystepper.step(25);
 lcd.setCursor(8,2);
 lcd.print(second());
 if (second() >= timelimit)
 break; }

 while (digitalRead(buttonBackward) == LOW
 && digitalRead(buttonRight) == HIGH && digitalRead(buttonLeft) == HIGH ) {
 Ystepper.step(-25);
 lcd.setCursor(8,2);
 lcd.print(second());
 if (second() >= timelimit)
 break; }

 digitalWrite(22,LOW); //disables hold (no current supplied to stepper when idle)
 digitalWrite(24,LOW);
 digitalWrite(26,LOW);
 digitalWrite(28,LOW);

 //--------------Z axis------------------------------------
 while(digitalRead(buttonUp) == LOW) {
 Zstepper.step(50);
 lcd.setCursor(8,2);
 lcd.print(second());
 if (second() >= timelimit)
 break; }

 while (digitalRead(buttonDown) == LOW) {
 Zstepper.step(-50);
 lcd.setCursor(8,2);
 lcd.print(second());
 if (second() >= timelimit)
 break; }

 digitalWrite(30,LOW); //disables hold (no current supplied to stepper when idle)
 digitalWrite(32,LOW);
 digitalWrite(34,LOW);
 digitalWrite(36,LOW);

 //----------------Claw servo------------------------------------
 val = analogRead(potpin); // reads the value of the potentiometer (value between 0 and 1023)
 val = map(val, 0, 1023, 0, 179); // scale it to use it with the servo (value between 0 and 180)
 myservo.write(val); // sets the servo position according to the scaled value
 delay(15); // wait for the servo to get there

 //------------------Print Time Expire--------------------

 }

} // master end