//+------------------------------------------------------------------+

//| CLAUDE.mq5 |

//+------------------------------------------------------------------+

#include <Trade/Trade.mqh>

input double LotSize = 0.1;

input int EMA\_Period = 9;

input int MaxSpread = 20;

input double RiskPercent = 2.0;

input int ATR\_Period = 14;

input double ATR\_SL\_Multiplier = 1.5;

input double ATR\_TP\_Multiplier = 2.5;

int EMA\_Handle;

int ATR\_Handle;

double EMA\_Buffer[];

double ATR\_Buffer[];

CTrade trade;

bool isInitialized = false;

int OnInit()

{

if(!TerminalInfoInteger(TERMINAL\_CONNECTED))

{

Print("Terminal not connected!");

return(INIT\_FAILED);

}

EMA\_Handle = iMA(\_Symbol, PERIOD\_CURRENT, EMA\_Period, 0, MODE\_EMA, PRICE\_CLOSE);

ATR\_Handle = iATR(\_Symbol, PERIOD\_CURRENT, ATR\_Period);

if(EMA\_Handle == INVALID\_HANDLE || ATR\_Handle == INVALID\_HANDLE)

{

Print("Error creating indicators");

return(INIT\_FAILED);

}

ArrayResize(EMA\_Buffer, 3);

ArrayResize(ATR\_Buffer, 2);

ArraySetAsSeries(EMA\_Buffer, true);

ArraySetAsSeries(ATR\_Buffer, true);

isInitialized = true;

return(INIT\_SUCCEEDED);

}

void OnDeinit(const int reason)

{

if(EMA\_Handle != INVALID\_HANDLE) IndicatorRelease(EMA\_Handle);

if(ATR\_Handle != INVALID\_HANDLE) IndicatorRelease(ATR\_Handle);

isInitialized = false;

}

double CalculatePositionSize(double stopLoss)

{

if(stopLoss == 0) return LotSize;

double accountBalance = AccountInfoDouble(ACCOUNT\_BALANCE);

double riskAmount = accountBalance \* (RiskPercent / 100);

double tickSize = SymbolInfoDouble(\_Symbol, SYMBOL\_TRADE\_TICK\_SIZE);

double tickValue = SymbolInfoDouble(\_Symbol, SYMBOL\_TRADE\_TICK\_VALUE);

if(tickSize == 0 || tickValue == 0) return LotSize;

return MathMin(riskAmount / (stopLoss \* tickValue / tickSize), LotSize);

}

void OnTick()

{

if(!isInitialized || !TerminalInfoInteger(TERMINAL\_CONNECTED)) return;

MqlTick last\_tick;

if(!SymbolInfoTick(\_Symbol, last\_tick)) return;

if(SymbolInfoInteger(\_Symbol, SYMBOL\_SPREAD) > MaxSpread) return;

if(CopyBuffer(EMA\_Handle, 0, 0, 2, EMA\_Buffer) <= 0) return;

if(CopyBuffer(ATR\_Handle, 0, 0, 1, ATR\_Buffer) <= 0) return;

bool isAboveEMA = last\_tick.bid > EMA\_Buffer[0];

bool wasAboveEMA = last\_tick.bid > EMA\_Buffer[1];

if(isAboveEMA != wasAboveEMA)

{

for(int i = PositionsTotal() - 1; i >= 0; i--)

{

if(trade.PositionClose(PositionGetTicket(i)))

Sleep(100); // Prevent overload

}

}

if(PositionsTotal() == 0)

{

double atr = ATR\_Buffer[0];

if(isAboveEMA && !wasAboveEMA)

{

double stopLoss = last\_tick.ask - (atr \* ATR\_SL\_Multiplier);

double takeProfit = last\_tick.ask + (atr \* ATR\_TP\_Multiplier);

double positionSize = CalculatePositionSize(last\_tick.ask - stopLoss);

trade.Buy(positionSize, \_Symbol, last\_tick.ask, stopLoss, takeProfit);

}

else if(!isAboveEMA && wasAboveEMA)

{

double stopLoss = last\_tick.bid + (atr \* ATR\_SL\_Multiplier);

double takeProfit = last\_tick.bid - (atr \* ATR\_TP\_Multiplier);

double positionSize = CalculatePositionSize(stopLoss - last\_tick.bid);

trade.Sell(positionSize, \_Symbol, last\_tick.bid, stopLoss, takeProfit);

}

}

}