# Önálló laboratórium 1 – Beszámoló

Devizakereskedő algoritmus fejlesztése

## Gurubi Barnabás – DXEXVR

### *Konzulens: Kövesdán Gábor*

## **Feladat röviden**

A hallgató feladata a devizakereskedés és a technikai elemzés megismerése és kereskedő algoritmus fejlesztése.

## **Célkitűzés**

Az elsődleges cél a témával való megismerkedés. Ennek első lépése a devizakereskedéssel világának felderítése. A devizakereskedés egyik elengedhetetlen eszköze a technikai elemzés, amelynek egyik előnye, hogy viszonylag könnyen algoritmizálható. A technikai elemzés eszközeinek és módszereinek megismerése után, ezek automatizálására szolgáló eszköztár kipróbálása és alapszintű elsajátítása volt a következő lépés. A végcél, az új ismeretek segítségével egy működő deviza kereskedő bot fejlesztése, ami egy adott döntés alapú stratégia alapján csinál kötéseket. Végül a bot eredményeinek feldolgozásával zárulnak a feladatok és célok.

## **Bevezetés**

Jelen beszámoló nem arra szolgál, hogy az összes felvett új ismeretet bemutassa, hanem sokkal inkább az önálló munka ismertetésével foglalkozik. A feladat első lépése a devizakereskedés alapfogalmainak és folyamatainak elsajátítása, valamint a technikai elemzés eszközeinek megismerése. Ennek a lépésnek csak egy összefoglalóját fogja tartalmazni a következő fejezet, olyan aspektusban, hogy honnan is milyen új ismeretek kerültek elsajátításra.

A második lépés ennek a kereskedésnek magának és az algoritmizálására szolgáló eszközök megismerése (*MetaTrader, MQL*). Ezekről csupán szintén összefoglaló, vázlatos jelleggel esik szó, hiszen hozzáadott önálló munkával nem szolgálnak.

Az utolsó lépés, döntésalapú bot(ok) kódolása, és kipróbálása. Bemutatásra kerül a megírt kódrészletek lényegi része, olyan szemlélettel, hogy azt egy fejlesztő megértse, és le tudja vonni a konklúziót.

Az adott részeknél az esetleges tanulságok is megjelennek.

## **Devizakereskedés és technikai elemzés**

A témában alapvetően két helyről szereztem tudást. Az egyik egy bróker ingyenes videókurzusa, ahol bemutatják a fontos alapfogalmakat röviden, bevezetnek ebbe a világba dióhéjban. A másik forrás egy a témában sok anyagot tartalmazó honlap írásos tananyaghalmaza volt[[1]](#footnote-1), ez szintén ingyenes. Utóbbi rendkívül sok anyagot tartalmaz, a nagyon alapoktól kezdve egyre mélyebben igyekszik bemutatni a téma minél több szegletét.

Az alábbi témák megismerésével foglalkoztam (mögötte jelölve 1-től 5-ös skálán az ismeretek mennyiségét, hogy kicsit összehasonlíthatóak legyenek a pontok):

* Forex piac felépítése és működése (4)
* Technikai elemzés alapjai, fogalmai (3)
* Chart fajták és ide kapcsolódó fogalmak (3)
* Különböző fajta indikátorok, egyszerű stratégiák, mérőszámok (5)

**Tanulság, vélemény:**

Bár általam létrehozott eredményt ez a rész nem adott, kifejezetten érdekesnek találtam a témát. Az írásos anyagok jól feldolgozhatóak voltak, és kellően sok van, nem kell félni, hogy kifogyunk belőlük. Fontosnak tartom utólag visszatekintve is a téma megismerését, mert a kódolás részben is erősen támaszkodnak a szakmabeli fogalmakra, ha ezekkel nem vagyunk tisztában, legalább jelentésileg, akkor jelentősen megnehezedik az egyes kódolási tutorial-ok, blogpost-ok feldolgozása.

## **MetaTrader és az MQL nyelv**

A **MetaTrader** egy rendkívül elterjedt eszköz melynek segítségével kereskedhetünk a FOREX piacon. Az előző fejezetben említett videókurzusban ennek egy-két funkciója is bemutatásra került. Lényeg, hogy ennek a programnak a segítségével történik minden művelet a kereskedés során. Ebből kifolyólag rendkívül sok funkcióval rendelkezik a program, én ezeknek csak egy kis részével foglalkoztam, ami főként a kereskedő botok (MetaTrader terminológiával *Expert Advisor*-ok) futtatásához és teszteléséhez szükséges.

Az **MQL** (*MetaQuotes Language*) nyelv a már említett *Expert Advisor*-ok, programozásához használatos programozási nyelv. Ennek két fő verziója az elterjedt, az *MQL4* és *MQL5*. Az előbbivel kezdtem az ismerkedést szintén egy videókurzus alapján. Ezt én saját tapasztalat alapján egy *C*-hez igen hasonlító nyelvként írnám, le. Egy új változat az *MQL5*, ami már sokkal inkább közeledik a *C++* nyelvhez, így már megjelenik benne az objektumorientáltság.

A videókurzuson bemutatott dolgokat így először *MQL4*-ben próbáltam ki, azonban ezeket fokozatosan megnéztem *MQL5-*ben is. Ez számomra kényelmesebb volt, így a továbbiakban ezt használtam és a végső, elkészült kódrészletek is ezzel a nyelvvel íródtak.

**Tanulság, vélemény:**

A használt *MQL* nyelv mindkét változata valamennyire azért kényelmetlen, főleg a magas absztrakcióval rendelkező modern nyelvekhez szokott szemmel nézve. A félév végére, azért az *MQL5*-tel megbarátkoztam, ennek segítségével azért már kellő hatékonysággal lehet dolgozni. Ezek alapján mindenképp az 5-ös verziót preferálom a 4-es felett.

## **Saját kereskedő bot**

A már említett *MQL4* alapú kurzus által bemutatott bot-ból indultam ki. Ez egy nagyon egyszerű stratégiát tartalmaz, ami röviden a következő:

* Minden nap egy adott időpontban (pl. reggel 9) lezárja az éppen nyitva lévő megbízásokat és megnézi, hogy az adott napi első árfolyamérték milyen viszonyban van a jelenlegivel
* Ha a jelenlegi magasabb, mint a nap eleji, akkor **vétel** történik, ha alacsonyabb akkor pedig **eladás**

Lényegében ennyi is a logika alapja. Természetesen emögött a logika mögött kevés a technikai elemzés elmélete, és nem is egy jövedelmező jól megalapozott logika. Felmerülhet a kérdés, hogy akkor miért ezt vettem alapul. A válasz az, hogy meglehetősen jól tesztelhetőnek találtam. Olyan tulajdonságokkal rendelkezik, amik könnyen ellenőrizhetők kezdő szinten. Itt arra gondolok, hogy minden nap történik alapvetően egy kötés, egy jól ellenőrizhető irányba.

Ezt az alap kiinduló logikát a **daily-system.mql** fájl tartalmazza.

### ***daily-system.mql***

A videókurzus implementációja *MQL4* alapú volt így ezt dolgoztam át *MQL5*-re. Alapvető funkciókkal rendelkezik, melyek egy része kívülről is paraméterezhető, ezáltal optimalizálható. Minden kötéshez tartozik *take profit* és *stop loss* szint.

Maga a logika természetesen az *OnTick()* metódusban található. Annak érdekében, hogy ezt az egyszerű logikát ne homályosítsák el a különböző technikai részletek (pozíció nyitása, zárása, stb…), kiemeltem ezeket segédmetódusokba, amik egyszerűen használhatóak.

Nyilvánvalóan ez a rendkívül alapvető stratégia nem produkál kifejezetten jó eredményeket, kiindulási alapnak tökéletes volt (egyszerűen megismerkedhettem az alapvető implementálandó funkciókkal).

Mint tudjuk ez a rendszer is rendelkezik *stop loss* funkcióval, hiszen ezt teljesen támogadja az *MQL.* Ennek azonban egy széleskörűen elterjedt okosabb változata a *trailing stop loss*, amely a sima változattal ellentétben a veszteség ellen nem egy fix védelmi szintet ad, hanem egy adaptívat. A következő célom egy ilyen funkció implementálása volt, oly módon, hogy az utána könnyen és sokféleképpen újrafelhasználható legyen a továbbiakban.

### ***TrailingStop.mqh***

Ennek során megszületett egy include-olható fájl, amiben létrehoztam egy ilyen funkciót, olyan módon, hogy az könnyedén bővíthető legyen a későbbiekben is. Szerencsére találtam egy kiváló blogpost-ot, amit felhasználhattam[[2]](#footnote-2). A célom kifejezetten az volt, hogy megértsem a működését a post-ban bemutatott rendszernek, így igyekeztem az alapján, de magam írni a sajátomat. Egyes bemutatott funkciókat kivettem, a kódot kissé átalakítottam. A rendszer felépítését és működését röviden bemutatom:

Az alap egy absztrakt osztály, ami a rendszer vázát definiálja – **TrailingStop**. Ez összefogja a különböző *trailing stop* számítási és kezelési módszerek közös részét, így ezekkel később kibővítve megkaphatjuk a konkrét stratégiákat, egyszerűen.

A kód konkrét részleteit itt nem írom le, hiszen az rendkívül feldúzzasztaná a dokumentációt, azonban a lényegi működést leírom.

Annak érdekében, hogy általános legyen a rendszer, így paraméterezhető, hogy milyen devizapáron, milyen időablakban működjön, valamint, hogy minden adatbeérkezés pillanatában lefusson, vagy csak az előző gyertyára.

Az ősosztály által megvalósított közös funkcionalitás a következő:

* Amikor lefut, azonosítja, hogy éppen milyen trendben van az árfolyam (az, hogy ez hogyan történik, ki van emelve, és a leszármazott osztályok valósítják meg).
* A trend alapján megnézi, hogy van-e annak megfelelő pozíció (*bull* trendhez vételi, hiszen ekkor ennek várható, hogy állítani kell a *stop loss*-át; *bear* trendhez pedig eladási)
* Ha van megfelelő pozíció az adott trend-hez, akkor kiszámolja, hogy a jelenlegi árfolyam viszonyok mellett mennyinek kellene lennie éppen a *stop loss*-nak (ezt szintén a leszármazott osztály bővíti ki)
* Amennyiben szükséges akkor módosítja a nyitott pozíció *stop* *loss*-át (vételinél felfele, eladásinál lefele)

Mint látjuk ez az absztrakt viselkedés könnyen specifikálható a leszármazottak által. Lényegében két pontot: **a trend meghatározásában** és **a következő stop loss érték kiszámításában**.

Két megvalósító osztályt implementáltam, egy *SAR* indikátor alapút (**ParabolicTrailingStop**) és egy *NRTR* indikátor alapút (**NRTRTrailingStop**).

Az előbbi a *Parabolic Stop And Reverse – SAR* indikátor segítségével működik, hiszen ez az indikátor meghatározza az éppeni trendet, amit felhasználva tud dönteni rendszer. Az új *stop loss* érték pedig a trendtől független az előző zárási ár lesz.

A *Nick Rypock Trailing Reverse – NRTR* indikátor a támasz és ellenállás (*support* és *resistance*) szintek segítségével szintén meg tudja állapítani a trendeket (és trend változást). Az új *stop loss* érték pedig a trendnek megfelelően egy támasz (bull) vagy pedig ellenállás (bear) lesz.

A konkrét implementáció megtekinthető a forráskódból, valamint a bemutató videón kicsit részletesebben átfutok rajta.

### ***daily-system-sar-sl-no-signal.mql***

Az implementált *trailing stop loss* rendszer *SAR* alapú változatának segítségével az eredeti alap logikát megtámogatva készült el az adott stratégia. Ez a konkrét döntést nem befolyásolja, azonban a veszteségeket remélhetőleg csökkenti. A beszámoló végén mellékelni fogok pár eredményt, amit visszatesztelés segítségével kaptam és látható lesz, hogy milyen hatással volt az eredeti rendszerre a hozzáadott *trailing stop loss* funkcionalitás.

### ***daily-system-nrtr-sl-no-signal.mql***

Az előzőhöz nagyban hasonló bot, azonban a *trailing stop loss NRTR* alapú változatát használja. A követekző lépés az, hogy a meglévő egyszerű logikát kicsit bővítsem. A bővítés alapja az lesz, hogy használja a *trailing stop loss* általi indikátort, vagyis az azáltal mondott trendet. Ha ez megegyezik az eredeti megállapítással, akkor nyisson csak új pozíciót.

### ***daily-system-sar-sl.mql***

Az említett módon, a vételi és eladási pozíció nyitásáról szóló döntéshez figyelembe vesszük a *SAR* indikátor által mutatott trendet. Ezzel megpróbálva kiküszöbölni, azt, hogy trend-el szemben ne nyissunk pozíciót.

### ***daily-system-nrtr-sl.mql***

Az előzőhöz hasonlóan támogatva van a döntés, azonban az *NRTR* indikátor alapján.

**Tanulság, vélemények:**

A kódolás során a *trailing stop loss* rendszer implementálását tartottam a legérdekesebbnek, hiszen ez egy olyan komponens, amit utána többször fel tudtam használni. Tanulságként elmondható, hogy nem a legbarátságosabb dolog *MQL* kódot írni, van, hogy sokáig kell hibákat keresgélni, de azért ilyen szinten még megoldhatóak a felmerülő problémák.

Az *MQL5*-öt már valóban egy alkalmazható, és használható nyelvnek tartom, főleg amiatt, hogy viszonylag egyszerűen létre lehetett, hozni benne egy ilyen komponenst.

A belefektetett energia szempontjából, szintén a *trailing stop loss* rendszer implementálása és csiszolgatása vette el a legtöbb időt.

## **Bemenő paraméterek optimalizálása**

Mint említettem az egyes botok rendelkeznek bemenő paraméterekkel, amiktől függ az eredményességük. Ilyen lehet pl. a *take profit* vagy *stop loss* mértéke, vagy akár a bemutatott adott óra kezdetén trade-elő stratégia időpontja. Ezeket megéri kiemelni és nem beleégetni a kódba, hiszen ekkor könnyen változtathatóak.

Azonban kívülről is ezeket kézzel átírogatni és mindig kipróbálni, hogy milyen eredmény ad problémás lenne. Szerencsére a **MetaTrader** biztosít optimalizálási funkciót, amellyel megadható, hogy mely paraméterek milyen kezdő értéktől milyen limitig (adott lépéssel) legyenek kipróbálva. Így rengeteg bemenő paraméter változattal tesztelésre kerül az *EA* (*Expert Advisor*), és végül megtudhatjuk, milyen paraméterezéssel volt a legeredményesebb a stratégiánk.

## **Eredmények**

A bemutatott *EA*-ok kipróbálására, tesztelésére szintén nyújt támogatást a **MetaTrader**. Segítségével valós adatokon egy múltbéli időintervallumon lefuttatható az *EA*, és így megláthatjuk, hogy milyen eredmény ért volna el, ha futott volna akkor. Természetesen a paramétereket optimalizálva (és a legjobbat kiválasztva).

Mivel napi trade-eket végző botokról van szó, így én az elmúlt év adatain próbáltam ki ezeket (H1-en, EUR/USD párral). Minden esetben 10 000 *USD*-s számla egyenleggel indultak a botok, és minden trade mérete 1 *lot* volt. Egy táblázatban összefoglaltam, hogy az év eleje óta mostanáig (tehát kb. 5 és fél hónap), milyen záró egyenleget hoztak össze az egyes *EA*-k (egyéb adatokkal kiegészítve).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EA neve** | **Trade száma** | **Össz nyereség** | **Össz veszteség** | **Végső egyenleg** |
| daily-system | 97 | 14 881 | 12 342 | 2 539 |
| daily-system-sar-sl-no-signal | 97 | 13 334 | 8 638 | 4 696 |
| daily-system-nrtr-sl-no-signal | 97 | 12 995 | 8 674 | 4 321 |
| daily-system-sar-sl | 33 | 4 904 | 1 944 | 2 960 |
| daily-system-nrtr-sl | 51 | 8 383 | 3 490 | 4 893 |

Nyilvánvalóan egy múltbéli sikeres teszt nem bizonyítja az élőben való sikeres kereskedést, hiszen előfordulhat, hogy az adott idő intervallumra optimalizáltan működik csak jól az adott stratégia. Azonban jó alapot ad a további tesztelésre, más intervallumokra, valamint demo számlán való élő használatra.

Én a jelenlegi eredményes *EA*-at nem próbáltam ki rendesen demo számlán élőben, hiszen ezek napi kötéseket végeznek, így bármi jelentős eredmény érdekében 1-2 hónapig futniuk kellene.

**Tanulság, vélemény:**

Nyilvánvalóan egy múltbéli sikeres teszt nem bizonyítja az élőben való sikeres kereskedést.

dsdfs ds d sd ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd ddd d d

Az előzőhöz hasonló bot, azonban a *trailing stop* az *NRTR* változatot használja. Az implementált *trailing stop loss* rendszer *SAR* alapú változatának segítségével az eredeti alap logikát megtámogatva készült el az adott stratégia. Ez a konkrét döntést nem befolyásolja, azonban a veszteségeket remélhetőleg csökkenti. A beszámoló végén mellékelni fogok pár eredményt, amit visszatesztelés segítségével kaptam és látható lesz, hogy milyen hatással volt az eredeti rendszerre a hozzáadott *trailing stop loss* funkcionalitás.

Ssssssssssss sssssssssssssss ssssssssssssssssssssssssssssssss ssssssssssssssssssssssssssssssss sssssssssss ssssssssss

1. https://www.babypips.com/learn/forex [↑](#footnote-ref-1)
2. https://www.mql5.com/en/articles/134 [↑](#footnote-ref-2)