Számítógépes Hálózatok

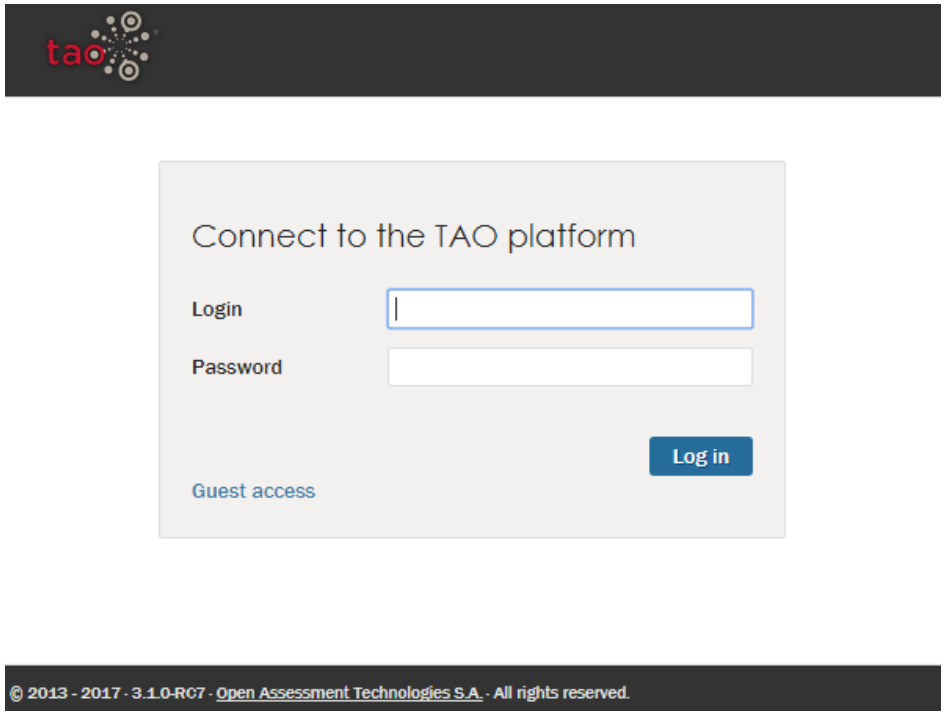
7. gyakorlat

Elérhetőségek

- honlap: http://szalaigj.web.elte.hu/
- email: szalaigindl@inf.elte.hu
- szoba: 2.507 (déli tömb)

Óra eleji kisZH

- Elérés:
  - https://oktnb16.inf.elte.hu

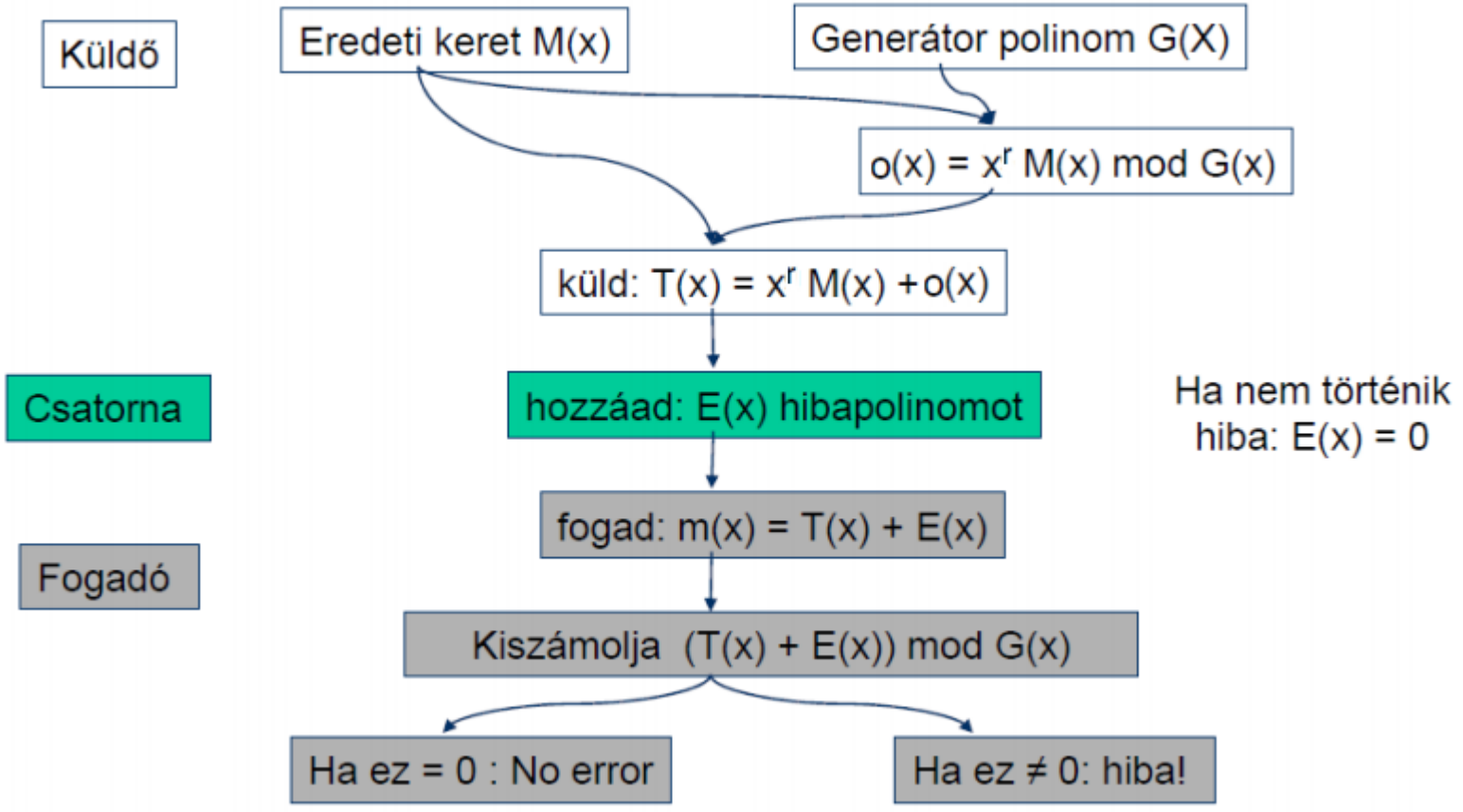


Gyakorlat tematika

- Hibajelző kód: CRC számítás
- Órai / házi feladat

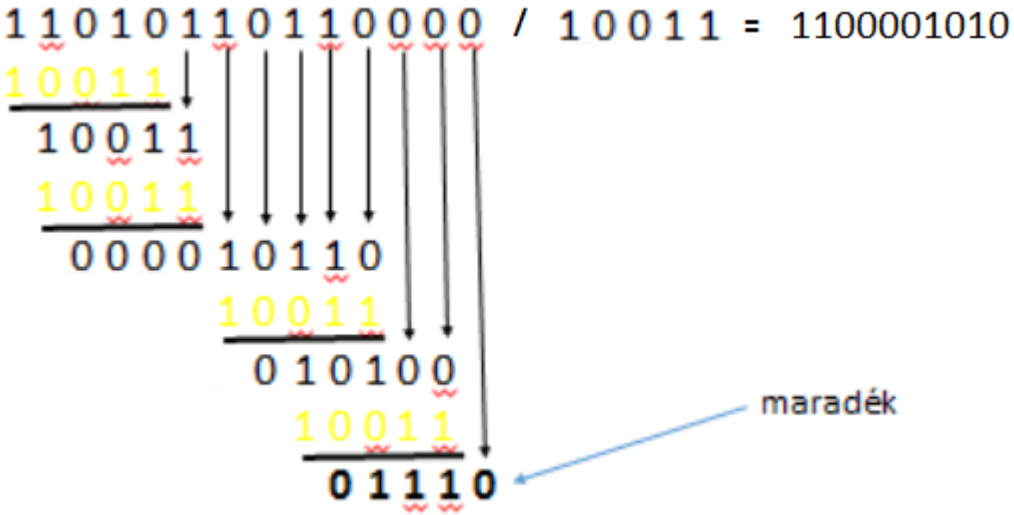
CRC hibajelző kód – emlékeztető

- Forrás: Dr. Lukovszki Tamás fóliái alapján



Példa CRC számításra – emlékeztető

- Keret ($M(x)$): 1101011011
- Generátor ($G(x)$): 10011
- Végezzük el a következő maradékos osztást: $\frac{1101011011}{10011}$
- (A maradék lesz a CRC ellenőrzőösszeg)



Ellenőrzés:

$$
\begin{array}{r}
110001010 \\
\cdot \quad 10011 \\
\hline
1100001010 \\
1100001010 \\
1100001010 \\
\hline
11010110111110 \\
+ \quad 01110 \\
\hline
11010110110000
\end{array}
$$

Példa CRC számításra – kiegészítés

- Az előbbi szorzásnál: 1100001010 megfelel a $1 \cdot x^9 + 1 \cdot x^8 + 1 \cdot x^3 + 1 \cdot x$, 10011 megfelel a $1 \cdot x^4 + 1 \cdot x + 1$ polinomnak
- Ha a polinom alakjukban összeszoroznánk:

$$(1 \cdot x^9 + 1 \cdot x^8 + 1 \cdot x^3 + 1 \cdot x) \cdot (1 \cdot x^4 + 1 \cdot x + 1)$$

$$= 1 \cdot x^{13} + 1 \cdot x^{10} + 1 \cdot x^9 + 1 \cdot x^{12} + 1 \cdot x^9 + 1 \cdot x^8 + 1 \cdot x^7 + 1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^5 + 1 \cdot x^2 + 1 \cdot x$$

$$= 1 \cdot x^{13} + 1 \cdot x^{12} + 1 \cdot x^{10} + \underbrace{(1+1)}_{0\ (mod\ 2)} \cdot x^9 + 1 \cdot x^8 + 1 \cdot x^7 + 1 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x$$

- Ez bináris alakban éppen a 11010110111110 lesz

Példa CRC számításra – kiegészítés

- Az előbbi osztásnál: 11010110110000 megfelel a $1 \cdot x^{13} + 1 \cdot x^{12} + 1 \cdot x^{10} + 1 \cdot x^{8} + 1 \cdot x^{7} + 1 \cdot x^{5} + 1 \cdot x^{4}$, 10011 megfelel a $1 \cdot x^{4} + 1 \cdot x + 1$ polinomnak
- Ha a polinom alakjukban végeznénk az osztást, akkor először az osztó legnagyobb fokú tagjával (x^4) leosztanánk az osztandó legnagyobb fokú tagját (x^{13})
- Ez x^9-et eredményezi. „Lekönyveljük”, és összeszorozzuk az osztóval: $x^9 \cdot (1 \cdot x^4 + 1 \cdot x +$

Példa CRC számításra – kiegészítés

- Ennek a polinomnak a legnagyobb fokú tagját osztjuk majd tovább az osztó legnagyobb fokú tagjával az előző lépésekhez hasonlóan, amely x^8-at eredményezi. „Lekönyveljük”, és összeszorozzuk az osztóval:

$$x^8 \cdot (1 \cdot x^4 + 1 \cdot x + 1) = x^{12} + x^9 + x^8$$

- Ezt kivonjuk az eredeti osztandóból, amely az $1 \cdot x^7 + 1 \cdot x^5 + 1 \cdot x^4$ polinomot eredményezi. Bináris alakban ez a polinom: 10110000. Ezt folytatjuk.
- A végén a „lekönyvelt” tagokat összeadjuk, amely az eredményt adja, továbbá a megmaradt, az osztónál kisebb fokú polinom a maradékot.

Feladat 1

- Adva a $G(x) = x^4 + x^2 + 1$ generátor polinom.
- Számoljuk ki $0110\ 1101$ bemenethez a 4-bit CRC ellenőrzőösszeget!
- Adjuk meg az átviteli üzenetet (a csatornára kerülő bitsorozatot)!

Feladat 1 megoldása

- Mivel a generátor polinom foka 4, ezért négy 0-t írunk a bemenet végéhez. A $G(x)$ bináris alakban: 10101 lesz, tehát a $\frac{0110\ 1101\ 0000}{10101}$ maradékos osztást kell elvégeznünk:

```
011011010000   /   10101
-----------------------
 11011010000
 10101
-----------
  1110010000
  10101
-----------
   100110000
   10101
-----------
    01100000
    10101
-----------
     110100
     10101
-----------
      11110
      10101
-----------------------
      01011   →   1011 a CRC ellenőrzőösszeg
```

Feladat 1 megoldása

- Az átviteli üzenet: 0110 1101 1011

Feladat 2

- Adva a $G(x) = x^4 + x + 1$ generátor polinom.
- Történt-e hiba az átvitel során, ha a vevő a következő üzenetet kapja:
  $1011\ 1001\ 1101\ 0111$

Feladat 2 megoldása

- A $G(x)$ bináris alakban: 10011 lesz, tehát a $\frac{1011\ 1001\ 1101\ 0111}{10011}$ maradékos osztást kell elvégeznünk, és ha nem nulla jön ki maradéknak, akkor hiba történt:

```
1011100111010111   /   10011
-----------------------------
1011100111010111
10011
---------------
 010000111010111
  10011
---------------
  0011111010111
   10011
---------------
   1100010111
   10011
---------------
    101110111
    10011
---------------
     01000111
      10011
-----------------------------
     001011      Nem nulla a maradék → hiba történt
```

Feladat 3

- Adva a $G(x) = x^4 + x^3 + x + 1$ generátor polinom.
- Számoljuk ki a $1100\ 1010\ 1110\ 1100$ bemenethez a 4-bit CRC ellenőrzőösszeget!
- A fenti üzenet az átvitel során sérül, a vevő adatkapcsolati rétege az $1100\ 1010\ 1101\ 1010\ 0100$ bitsorozatot kapja. Történt-e olyan hiba az átvitel során, amit a generátor polinommal fel lehet ismerni? Ha nem, akkor ennek mi lehet az oka?

Feladat 3 megoldása

- Mivel a generátor polinom foka 4, ezért négy 0-t írunk a bemenet végéhez. A $G(x)$ bináris alakban: 11011 lesz, tehát a $\frac{1100\ 1010\ 1110\ 1100\ 0000}{11011}$ maradékos osztást kell elvégeznünk:

```
1100101011011000000   /   11011
-----------------------------------
1100101011011000000
11011
-----------------------------------
 0010010111011000000
   11011
-----------------------------------
  1001111011000000
  11011
-----------------------------------
  100011011000000
  11011
-----------------------------------
  10101011000000
  11011
-----------------------------------
  1110011000000
  11011
-----------------------------------
  011111000000
  11011
-----------------------------------
  0100000000
  11011
-----------------------------------
  10110000
  11011
-----------------------------------
  1101000
  11011
-----------------------------------
  000100   →   0100 a CRC ellenőrzőösszeg
```

Feladat 3 megoldása

- Az előbbi számításnál az jött ki, hogy 1100 1010 1110 1100 0100 lenne az a bitsorozat, amelyet a vevő kapna. Ha ebből kivonjuk az alfeladatban megadott sorozatot, az alábbi eredmény jön ki:

$$
\begin{array}{rl}
 & 11001010111011000100 \\
- & 11001010110110100100 \\
\hline
 & 00000000001101100000
\end{array}
$$

- Tehát a két bitsorozat pontosan a generátor polinom többszörösével tér egymástól, amely tehát a hiba polinom. Ezt pedig nem lehet felismerni.

Feladat 4

- Készítsünk egy ***kliens***-***proxy***-***szerver*** alkalmazást, ahol:
  - a ***szerver*** egy UDP szerver,
  - a ***proxy*** a ***szerver*** irányába egy UDP kliens, a ***kliens*** irányába egy TCP szerver,
  - a ***kliens*** egy TCP kliens a ***proxy*** irányába
- Folyamat:
  - a ***kliens*** küldje a ‚Hello Server' üzenetet a ***proxy***nak,
  - amely küldje tovább azt a ***szerver***nek,
  - amely válaszolja vissza a ‚Hello Kliens' üzenetet a ***proxy***nak,
  - amely küldje tovább azt a ***kliens***nek

Órai / házi feladat

CRC

- Legyen tetszőleges számú ***küldő***, egy ***csatorna szerver*** és egy ***fogadó***
- Adott a generátor polinom: $G(x) = x^4 + x^3 + x + 1$, amelyet mindenki ismer
- Egy ***küldő*** indításkor csatlakozik a ***csatorna szerver***hez
- A ***csatorna szerver*** egy időben figyel több socketet
  - A bejövő kapcsolódásokra és a meglevő kapcsolatokból való olvasásra is → használjunk **select** függvényt!

Órai / házi feladat

CRC

- A ***küldő*** oldalon folyamatosan megadhatunk egyszerű 0-1 bitekből álló üzeneteket,
  - amelyre az alkalmazás a 4-bit CRC ellenőrzőösszeggel ellátott átviteli üzenetet számolja ki,
  - és küldi tovább a ***csatorna szerver***nek

Órai / házi feladat

CRC

- A ***csatorna szerver*** ciklikus módon váltakozva az alábbi hiba polinomokat adja az átviteli üzenetekhez:
  1. Nincs hiba polinom
  2. Van hiba polinom, és **nem** többszöröse a generátor polinomnak
  3. Van hiba polinom, de többszöröse a generátor polinomnak
  - (Megjegyzés: ezek a hiba polinomok lehetnek előre adottak)
- Azután továbbítja UDP protokoll használatával a ***fogadó***nak a hibával terhelt üzenetet
  - Figyelem! Ehhez még egy socketet definiálni kell egy új porttal a ***csatorna szerver***nél (és datagram típusú legyen ez a socket)
  - Itt a ***csatorna szerver*** legyen az UDP kliens, és a ***fogadó*** legyen az UDP szerver (azaz a ***fogadó***nál történjen meg a bind függvény hívása)

Órai / házi feladat

CRC

- A ***fogadó*** egy végtelen ciklusban folyamatosan várja a beérkező üzeneteket
  - A socketjén definiáljunk timeoutot (pl. 1 másodperccel), hogy le lehessen állítani, akkor is, ha nem jön üzenet a ***csatorna szerver***től
  - Így socket.timeout kivétel fog jönni, de azt kezeljük le úgy, hogy continue (vagy egy üres pass utasítás) szerepeljen a kivételkezelésnél
- A ***fogadó*** megpróbálja leellenőrzi a generátor polinom segítségével bejövő üzenetre, hogy történt-e hiba, és kiírja a kimenetre az eredményt
  - (Nyilván minden harmadik üzenetre tévesen azt a következtetést fogja levonni, hogy nincs hiba, hiszen abban az esetben a hiba rejtve marad)

Órai / házi feladat

CRC

- Természetesen ne használjunk a CRC számításhoz mások által megírt CRC számító modult! Ennek implementálása is a feladat része!
- Mivel CRC számítást a ***küldő*** és ***fogadó*** is végez, ezért érdemes ezt a funkcionalitást kiszervezni egy külön modulba
- Ehhez segítség: ha a `haziCRCCalculator.py` tartalmazza a szükséges funkcionalitást pl. `calculateRemainder(...)` néven, akkor a többi forrásfájlba egyszerűen az alábbi módon tudunk erre ráhívni (ha ugyanabban a könyvtárban vannak):

```
import haziCRCCalculator
...
haziCRCCalculator.calculateRemainder(...)
```

Órai / házi feladat

CRC

- Segítség a CRC számításhoz (de nem kötelező ezt használni):
- Pythonnál az int(str1,base2) konverzió a bemeneti str1 sztringből a megadott base2 alapján, - amely a számrendszert jelöli, - egész számot készít:

```
>>> i1 = int('1011100111010111',2)
>>> i1
47575
>>> i2 = int('1001100000000000',2)
>>> i2
38912
```

- Bitenkénti műveletek: <<, >>, &, |, ~, ^, pl. a bitenkénti XOR:

```
>>> i1^i2
8663
```

- Az előbbi sztring reprezentációja:

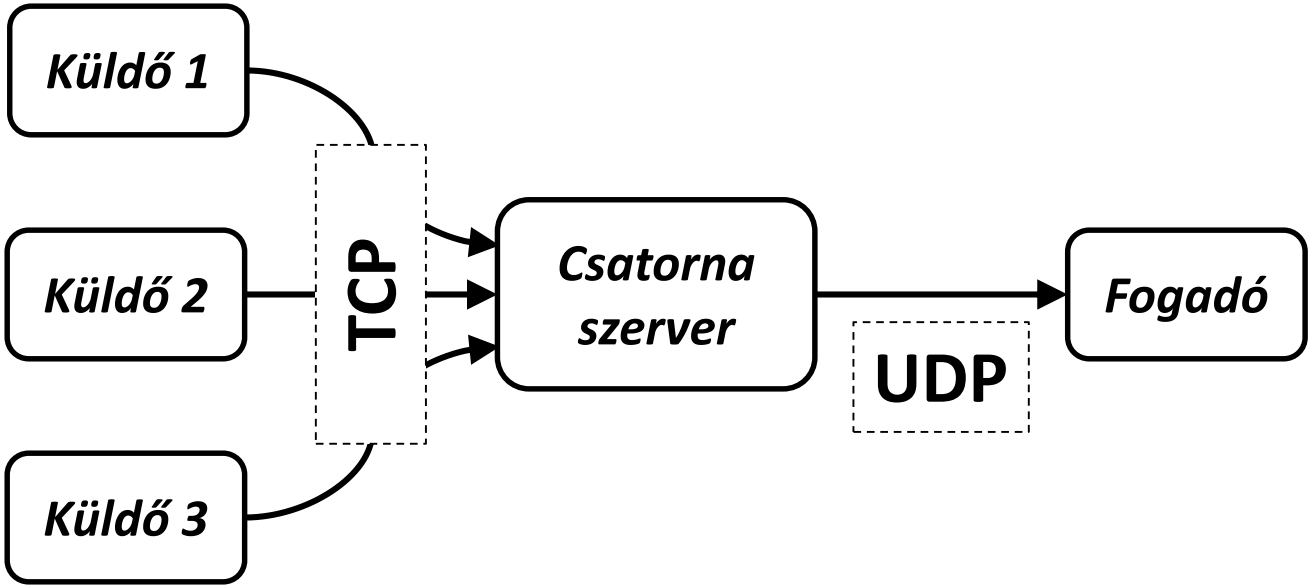
```
>>> "{0:b}".format(i1^i2)
'10000111010111'
```

Órai / házi feladat

CRC

- Ennek a házi feladatnak a pontozása (kivételesen):
  - Mindenképpen működnie kell a ***küldő-csatorna szerver*** kommunikációnak (több ***küldő***t ki tud egyidejűleg szolgálni **select** használatával), illetve a CRC számításnak (eddig ez 1 pont) továbbá:
  - Ha a ***csatorna szerver*** oldalon a hiba polinom hozzáadása megfelelően működik → 1 pont
  - Ha van ***fogadó*** (a ***csatorna szerver***rel UDP kapcsolat használatával) és a hiba ellenőrzés megfelelően működik → 1 pont
- Tehát 1 extra pontot lehet szerezni erre a házira, ha valaki mindegyik részét megfelelően implementálja
  - (mivel egy jól megoldott házira alapvetően 2 pont jár)

Órai / házi feladat



```
e:\Gyak07>python haziReceiver.py
110010101001
110010101001
1101110000000
------------
   100101001
   110110000
------------
    10011001
    11011000
------------
     1000001
     1101100
------------
      101101
      110110
------------
       11011
       11011
------------
           0
No error
111011011000
111011011000
1101110000000
------------
  1101011000
  1101100000
------------
      111000
      110110
------------
        1110
error
111110100101
111110100101
1101110000000
------------
  1000100101
  1101100000
------------
   101000101
   110110000
------------
    11110101
    11011000
------------
      101101
      110110
------------
       11011
       11011
------------
           0
No error
```

```
e:\Gyak07>python haziChannelServer.py
new connection from ('127.0.0.1', 6887)
new connection from ('127.0.0.1', 6888)
received "110010101001" from ('127.0.0.1', 6888)
received "011101101111" from ('127.0.0.1', 6887)
received "111110111110" from ('127.0.0.1', 6888)
```

```
e:\Gyak07>python haziSender.py
<message> 0111011011
<message>
```

```
e:\Gyak07>python haziSender.py
<message> 11001010
<message> 11111011
<message>
```

VÉGE
KÖSZÖNÖM A FIGYELMET!