# Development of an Autonomous Ground Vehicle for the RobonAUT 2014 Contest

Csorvási Gábor Fodor Attila
*Department of Automation and Applied Informatics*
*Budapest University of Technology and Economics*
{csorvagep, attila.fodor.89}@gmail.com

**Abstract.** RobonAUT is a local robot development contest of the Faculty of Electrical Engineering and Informatics of BME. This paper describes the development process of an Autonomous Ground Vehicle which was one of the successful competitors of the 2014 contest. During software development of the embedded control system the possibilities of rapid prototyping have been exploited extensively. The paper demonstrates the power of model based design of control systems and embedded software, and describes the rapid deployment method in detail, which was used to integrate the MATLAB model with FreeRTOS on the target hardware.

**Keywords:** RobonAUT; Autonomous; AACS Workhsop; MATLAB; Real-Time; Control Systems; ...

## 1 Introduction

A dokumentum abból a célból jött létre, hogy segítséget nyújtson `STM32F4-Discovery` board alapú projektek fejlesztéséhez. A szükséges lépések nagyrészét igyekszünk bemutatni valós alkalmazások segítségével, melyeknek túlnyomó többsége a 2014-es **RobonAUT** versenyre történő felkészülés közben került kivitelezésre. A problémák számunkra is újak voltak, így nem ígérhetjük, hogy a legjobb megoldásokat fogjuk prezentálni itt, de a bemutatott eljárásról kijelenthetjük, hogy hasznosnak és megbízhatónak bizonyult.

Az írást három fő részre osztottuk, először bemutatásra kerülnek a `MATLAB-Simulink` modell alapú tervezés lépései és a fordítható kód generálása. Ezután a generált kód integrálására és élesztésére térünk rá `FreeRTOS` operációs rendszer alatt, legvégül pedig valós gyors prototípustervezési technikák kerülnek bemutatásra, a `MATLAB` közvetlen `STM32F4-Discovery` hardware támogatását felhasználva.

A **RobonAUT** verseny során egy olyan robot szoftverét és hardverét kell elkészíteni, amely lehetővé teszi a távirányítós autóból átalakított platformot, hogy egy ismeretlen pályán végighaladjon, és ott akadályokat teljesítsen, a szabályzatban meghatározott módon. A továbbiakban csak a szoftver környezettel foglalkozunk és feltételezzük, hogy megfelelő logikai jelek zajjal terhelten bár, de rendelkezésre állnak a szenzorokból, illetve az autó várakozásainknak megfelelően reagál a kimeneti jelekre, pl. a kormányszög beállítására.

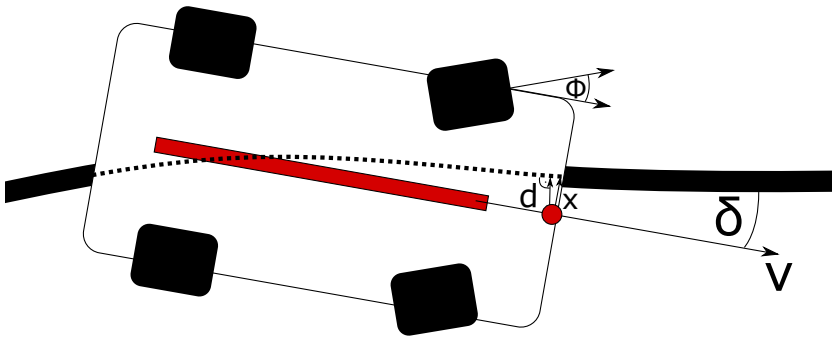


Figure 1: A probléma sematikus ábrája felülnézetből

A dokumentumban próbáljuk a szakirodalomban elterjedt jelrendszert alkalmazni, de az alábbi táblázat segítségével magyarázzuk a rendszeresen előforduló jelöléseket és összefüggéseket. Amennyiben valami hiányzik innen, az a szövegkörnyezetben kerül definiálásra.

| | |
|---|---|
| d | Pozícióhiba: Az első optikai szenzorsor középpontjának távolsága a vonaltól |
| $\delta$ | Szöghiba: Az első optikai szenzorsorra merőleges egyenes (középvonal) és a pályavonal érintője által bezárt szög |
| $\Phi$ | Kormányszög: A kerekek síkjainak a középvonallal bezárt szögeinek átlaga, Ackermann-kormányzás szerint |
| $\kappa$ | A pálya pillanatnyi görbülete $(1/R)$ |
| x | Vonalpozíció: az első optikai szenzorsor által érzékelt vonalak középpontjának előjeles távolsága a szenzorsoron a közepponttól mérve. |
| c | Vonalsebesség: x első deriváltja, a vonalpozíció mozgásának sebessége |
| v | Sebesség: Az autó pillanatnyi sebessége a középvonal mentén |

# 2 Integration

## 2.1 A generált kód felépítése

Amennyiben compact code placement beállítással generáltuk le a fileokat, csupán 4 számunkra hasznos file keletkezik:

- "subsystem neve".c
- "subsystem neve".h
- rtwtypes.h
- ert_main.c

A továbbiakban feltételezzük, hogy a generált subsystem neve "Controller" volt, az egyszerűbb olvashatóságért.

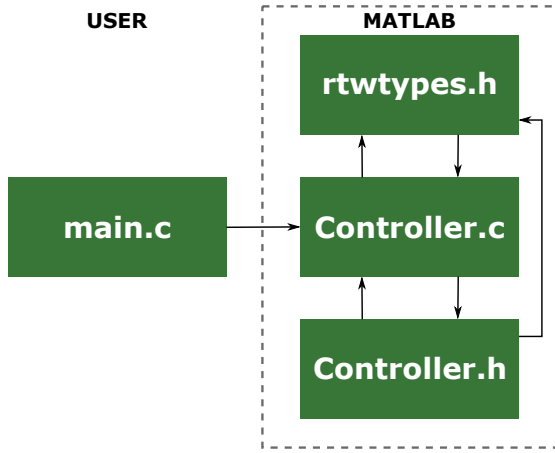


Figure 2: A generált fileok kapcsolata

A **Controller.c** és **Controller.h** fileokban van definiálva a rendszer működése. A kódból hozzávetőlegesen kiolvasható a modell működése. A **rtwtypes.h** fileban kerülnek definiálásra a rendszer által használt típusok, az **ert_main.c** pedig bemutatja a használatukat.

## 2.2 Simulink modell kezelése C kódban

A generált kód tökéletes megfelelője a Simulinkben futtatott modellnek. A generált fileok egy objektumot definiálnak, saját típusokkal és tagfüggvényekkel.

A rendszer inicializálását a Controller_initialize() függvénnyel tehetjük meg. Ez a modell futásának elídításának felel meg, beállítja a 0 időpillanatot a rendszerben, a kimenetek felveszik az alapértelmezett értéküket. A rendszerrel csupán a be és kimeneti interfészen keresztül kommunikálhatunk, valamint hatást gyakorolhatunk rá a `Step` függvény meghívási idejének változtatásával (ezt nem részletezzük, és ne is nagyon erőltessük). A bemeneti interfészt a Controller_U struktúra implementálja, ennek a változónak a tagjai felelnek meg egy-egy bemeneti jelnek.[1]

Miután beállítottuk a bemeneteket, a Controller_step() függvény meghívásával **léptethetjük** a modellt. A lépés megegyezik a generált rendszer **1 mintavételi idejű** lépésével. A Step függvény lefutása közben frissülnek a rendszer belső állapotai, valamint Controller_Y-ban a kimenetek. A kimenetet hasonlóléppen állítja elő a rendszer, Controller_Y struktúrában tárolva.

A bemenetek megadása, léptetés, kimenet kiolvasása az az elemi lépéssorozat, melyet rendszeresen **időzítve** végrehajtunk. Ha a modellnek időtől függő belső

[1] Többdimenziós jel esetén tömbként történik az átadás. 2 vagy több dimenzió esetén ne feledjük, hogy a MATLAB **oszlopfolytonosan** kezeli a tömböket.

állapotai is vannak[2], **pontos időzítéssel** kell biztosítanunk a periodikus meghívást. Ha a rendszer túlnyomó része MATLAB-ban készült, ez csupán timer időzítővel is biztosítható, ám ha más feladatokat is el kell látnia a rendszernek, valószínűleg egy Hard Real-Time operációs rendszerre is szükségünk lesz.

Rögtön felhívnám a figyelmet, ha eddig nem vált volna nyilvánvalóvá, hogy a generált forrásfileok közül **csak** az ert_main.c-t szabad módosítani. Ha úgy érezzük, hogy a többi generált kódba kell kézzel belenyúlni, akkor valamit elrontottunk. Továbbá meg lehet találni a megfelelő megfeleltetéseket a generált kód és a simulink modell között, de ez többnyire felesleges és **megbízhatatlan**. A legjobb fekete dobozként tekinteni a rendszerre, ha pedig valamilyen belső változóra szükség van debuggoláshoz, vegyük a fáradságot és vezessük ki kimenetre[3].

### 2.3 Példarendszer integrációja periodikus meghívással

A példarendszerünk bemenete a vonal pozíciója, kimenete pedig a kívánt kormányszög. A főprogramunkból a következőképpen tudjuk meghívni a modellt:

```
/* Pass inputs */
Controller_U.Position = line_position;

/* Update model */
Controller_step();

/* Receive outputs */
servo_position = Controller_Y.Servo;
```

Ezt a kódot kell egy olyan függvénybe beletennünk, melynek tudjuk biztosítani a periodikus meghívását.

A vonal pozíciót feldolgozhatjuk C-ben is akár, de minek, ha úgyis köré építünk egy MATLAB rendszert? Ehhez pointerrel tudjuk átadni az adattömböt, Simulinkben pedig vektor bemenetet kell beállítanunk.

## 3 Hardware-támogatás

A MATLAB 2013b verziójától elérhető direkt hardware-támogatás az STM32F4 Discovery fejlesztőkártyához, melyet a MATLAB Hardware Support oldaláról tölthetünk le. A támogatás segítségével villámgyorsan tesztelhetjük az elkészített szoftvert, soros kábel segítségével pedig akár Processor-in-the-Loop tesztet is végrehajthatunk[**?**, ].

[2] Ez minden esetben igaz, P szabályzónál bonyolultabb irányítás esetében

[3] A Goto/From simulink blokkpárral ezt elegánsan megtehetjük, a vezetékek összekuszálása nélkül

## 3.1 Gyors Prototípustervezés

A RobonAUT elsősorban gyors prototípustervezési munkát igényel. Rövid idő alatt, minél hatékonyabb párhuzamosítással és a lehető legkevesebb teszteléssel kell sok funkcióval rendelkező, többé-kevésbé megbízható rendszert építeni. A magas és az alacsony szintű irányítás fejlesztése teljesen párhuzamosan zajlik, ha megfelelően kiaknázzuk a lehetőségeket.

**Példa** Készítsünk Simulinkben egy LED-villogtató programot 5 perc alatt!

Miután feltelepítettük a support package-et[4], hozzunk létre egy új Simulink modellt, majd húzzuk be az **Embedded Coder Support Package for STMF4-Discovery Board** library-ből a szükséges blokkokat. Az ábrán látható módon állítuk össze a kapcsolást, majd konfiguráljuk a modellt az 1. részben leírtakhoz hasonlóan, de most a **Code Generation** menüben **Target Hardware**-nek válasszuk az `STM32F4-Discovery`-t, valamint ne csak kódot generáljunk (Pipa ki a **Generate code only** checkbox-ból).[5]

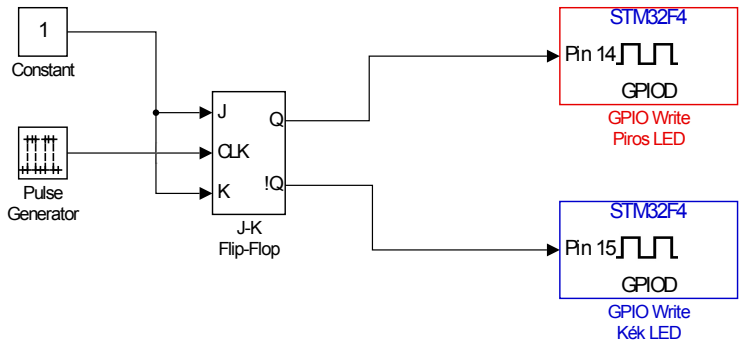


Figure 3: LED villogtató Simulink-modell

A teljes rendszert a **Code** menü **C/C++ Code** menüpontjából tudjuk buildelni. Ha mindent jól csináltunk, a jutalmunk a végén egy `.hex`, egy `.elf` és egy `.bin` file lesz a Working Directory-ben. Ezután az `STM ST-LINK Utility` segítségével felprogramozhatjuk a kártyát a generált fileok segítségével. Természetesen nem csak a ledeket tudjuk villogtatni, hanem az összes GPIO-t, gombot és analóg I/O-t kezelni tudjuk tetszőlegesen bonyolult modell köré építve. A GPIO Read és Write blokkok beállításához az `ST-Microelectronics` megfelelő segédletet nyújt[**?**].

[4] Ekkor indul a stopper

[5] Cheat sheet újoncoknak: A `Constant` blokk signal type-ját `boolean`-re, a `Pulse Generator` Pulse type-ját pedig `Sample based`-re kell állítani. A blokkokról pedig senki sem tudja, hogy melyik almenüben vannak, érdemes használni a keresőt. Ne felejtsük el a GPIO blokkok konfigurálását sem! Ha semmiképpen sem akar működni, akkor a kész modell letölthető `innen`.

Hasonló elv alapján épült fel egy standard servo jeleket feldolgozó projekt is, melyet egy távirányítós autó végfokozatának PWM-es vezérlésére használtam.A Simulink modell szintén letölthető `innen`, és mélyebb betekintést ad a modell alapú tervezés nyújtotta lehetőségekbe. Egy ilyen feladat elkészítése is inkább munkapercekben, mint -órákban mérhető. Korábban említettük, hogy akár PIL tesztelésre is lehetőség van. Ebben a dokumentumban erre az alkalmazási területre nem térünk ki, de egy későbbi bővített kiadásban előfordulhat, ha igény mutatkozik a témára.

# 4 Konklúzió

[1] Bízunk benne, hogy sikerült meghozni a kedvet az `STM32F4-Discovery` fejlesztőkártya kreatív használatához, és népszerűsíthettünk egy olyan fejlesztési irányzatot, ami a rendszerszemléletet helyezi a végeláthatatlan kódolás elé.

# Acknowledgments

The author would like to express his thanks to István Vajk [6] for his support as a scientific advisor. This work has been supported by the ... [7]

# References

[1] W. Weinrebe, A. Kuijpers, I. Klaucke, and M. Fink, "Multibeam bathymetry surveys in fjords and coastal areas of west-greenland," *AGU Fall Meeting Abstracts*, p. A1152, Dec 2009. Provided by the SAO/NASA Astrophysics Data System.

[6] Please mention the name of your advisor in the Acknowledgements section.
[7] Please mention the institution or organization that has supported your research work.