Skywalker 0.1

Anders Ydremark
Joakim Kvarnström

Inledning

Skywalker är ett programmeringsspråk skapat för att lätt kunna styra motorer och servon. Tanken är att man ska kunna skriva rutiner för att kunna får motorer och servon att arbeta och bete sig på olika sätt under en viss tid. För att kunna skriva en rutin måste man först skapa ett interface där man deklarerar de motorer och servon som ska användas.

```
interface Main
        Motor = {rightEngin,leftEngin}
        Servo = {rudder, aileron, elevator}
end
```

För att ett interface ska skapas och fungera måste man först skapa en ruby-fil med data om de olika motorerna och servona i. Ett exempel på hur denna fil kan se ut är:

```
Motor1 = mainEngine
ID = 1
maxThrust = 100
minThrust = 0
thrust = 0

Servo1 = elevator
ID = 2
maxPull = 50
minPull = -50
pull = 0
```

När detta interface är skapat kan man komma åt exempelvis “rightEngin” genom att skriva Motor[1]. Eller “elevator” genom Servo[3]. När man sedan har skapat ett Main interface kan man skapa en Main rutin. Varje program måste ha en Main rutin. Mer om detta senare. Skapa Main rutin:

```
routine Main
        Motor[1] = 10
        wait(5)
        Servo[2] = 90
        wait(10)
end
```

I denna rutin Sätter vi motor 1 (rightEngin) till 10. Vi väntar sedan i 5 sekunder innan vi sätter Servo 2 (aileron) till 90. Därefter kommer ruitinen vänta 10 sekunder innan den stoppas. Exempel på hur problem kan lösas finns nedan.

Uppgift 1:

Lasse vill bygga ett system med en motorer och ett servo. Tanken är att motorn ska styra ett band som det kan åka paket på. när motorn sätts till max (100) så tar det 5 sekunder för paketet att åka från början till slutet. Lasse vill dock att när paketet har kommit halvvägs så ska en servoarm peta av paketet från bandet. Lasse har upptäckt att motorn drar för mycket batteri om man kör det på max så han vill köra det på ungefär 75% av max. Han har också listat ut att servoarmen måste ställas på max för att paketet ska kunna petas ned. Man kan anta att det kommer ett nytt paket på bandet varje gång ett paket har petats ned.

Lösning Uppgift 1:

```
interface Main
        Motor = {rullband}
        Servo = {arm}
end

routine Main
        Motor[1] = 75
        while (true)
                wait(5*0,75/2)
                Servo[1] = 100
                wait(0.5)
                Servo[1] = 0
        end
        Motor[1] = 0
end
```

Uppgift 2:

Nils Pärson har köpt en motor på rean. Nils har kopplat en axeltill motorn. T.ex. så har Nils kopplat motorn till ett kugg som är fäst i en axel som i andra änden har ett lottohjul. Kugget på Nils motor har 30 tänder och kugget som sitter på axeln har 40 tänder. Nils har listat ut att kugget på axeln måste snurra med en hastighet av 75 rpm för att lottohjulet ska snurra med rätt hastighet. Det tar i detta fall 1 minut och 32 sekunder för lottohjulet att snurra ett varv. Nils vill att när lottohjulet har snurrat ett varv så ska det vända och snurra ett halvt varv tillbaka. När detta är klart vill han att det ska snurra en kvarts varv frammåt igen. När detta är klart ska lottohjulet stanna. Du kan anta att 1 rpm = 1 % av max på motorn.

Lösning Uppgift 2:

```
Interface Main
        Motor = {lotteryWheel}
end

routine Main
        var speed = (40/30) * 75
        var time = 92

        Motor[1] = speed
```

```
        wait(time)
        Motor[1] = speed * -1
        wait(time/2)
        Motor[1] = speed
        wait (time/4)
end
```

Grammatik / Regler

Grunden i ett Skywalker-program är Main-funktionen, eller Main-rutinen som vi kallar den. Den definieras först med ett interface. I interfacet deklareras alla motorer, servon etc. som skall användas. Main-rutinen är den rutinen som kommer att kallas på först. Den kan, om det behövs, kalla på andra rutiner om man vill abstrahera kod som ska köras flera gånger. Andra rutiner behöver inget interface då alla komponenter som ska styras redan definierats i Main-interfacet.

För att kunna komma åt motorer och servon mm. behöver dessa definieras i en extern ruby-fil som sedan inkluderas. Det som definieras i den kommer man att kunna komma åt i interfacet.

De datatyper som kan användas i språket är:
Integer (Heltal)
Float (Flyttal)
Boolean (Sanningsvärden)

De kontrollsatser som finns är:
While-satser
If / else -satser

De operatorer som finns är:
= Tilldelningsoperator
+ Additionsoperator
- Subtraktionsoperator
* Multiplikationsoperator
/ Subtraktionsoperator

Där * och / går före + och -
Ordningen för + och - samt * och / är vänster först.

De typer av variabler som går att tilldela är heltal och flyttal och deklareras på följande sätt:
var variabel = 1337 eller
var variabel = 13.37

Variabler för booleska värden deklareras i interfacet och värdet sätts av externa komponenter.

<program>	::=	<interface_def> <routine_def>
<routine_def>	::=	“Routine” <identifier> <stmt_list> <terminator>
<interface_def>	::=	“Interface”<identifier> <interface_stmt_list> <terminator>
<interface_stmt_list>	::=	<interface_stmt> | <interface_stmt_list> <interface_stmt> | <stmt_list>
<stmt_list>	::=	<stmt> | <stmt_list> <stmt>
<stmt>	::=	<assign_stmt> | <while_stmt> | <if_else_stmt>
<interface_stmt>	::=	<interface_assign_stmt>
<interface_assign_stmt>	::=	“Motor” “=” “{“ <identifier_list> “}” | “Servo” “=” “{“ <identifier_list> “}”
<assign_stmt>	::=	“var” <identifier> “=” <addition>
<addition>	::=	<multi> | <addition> <addition_oper> <multi>
<multi>	::=	<primary> | <multi> <multi_oper> <multi>
<primary>	::=	“(“ <addition> “)” | <atom>
<terminator>	::=	“end”
<identifier>	::=	[a-zA-Z]
<identifier_list>	::=	<identifier> | <identifier_list> <identifier>
<addition_oper>	::=	“+” | “-“

<multi_oper>	::=	“*” | ”/”
<atom>	::=	<integer> | <float> | <identifier>
<while_stmt>	::=	“while” <bool_expr> <stmt_list> <terminator>
<boolean>	::=	“true” | “false”
<bool_expr>	::=	<addition> <rel_oper> <addition> | <boolean>
<if_else_stmt>	::=	“if” <bool_expr> <stmt_list> <terminator>
<rel_oper>	::=	“= =” | “<” | “>” | “>=” | “<=”
<integer>	::=	[0-9]
<float>	::=	[0.00-9.99]