

! " # \$ % & ' ( ) \* ' + , % - " . / ' 0 ' 1 2 " &  
\* 3 4 3 \$ 3 & # 5 % & 6 RHex

Yarmolinskiy Arseniy

Version 0.1.0, 14.06.2024

# Table of Contents

<+%>%- " % .....	1
! 5%&' ( ) *' +, %- " . .....	2
E#\$3; - " 9" .....	8

! " # \$ % & #

7' - - 68 ) \*3%9\$ . +, . %\$#. ) \*3438 ) %\* ' + ( ) \* ' +, %- " " / % # \$ " - 30" & \*343\$3& ": ' ) ; " 9". <%#=  
" #53>- 68 93>) \*3%9\$' &3? - 3 - ' 8\$" + ) (4, " ; - 3& \*%) 3@" \$3\*" " [1].

! " # \$ % & ' ( ) \*

' (#) \* +, - \* " . # % & /

7, . >+ " ? %- " . \* 343\$' >, . 9' ? >38 , ' ) 9" 3) \*%>%, . %\$#. ? %, ' %&3% \$%9(2%% ) 3, 3? %- " % +  
' + " #" &3#\$" 3\$ \$\*' %9\$3\*" " >+ " ? %- " . " \$%9(2%8 A' @6 B" 9, 30\*' &&6 #33\$+%\$#\$+(1 2%8  
, ' ) 9" . < > ' , =- %8 / %& C\$3\$ (03, 3\$\*' 4' \$6+ ' %\$#. / %#\$=1 , 39' , =- 6&" D-DE \*%0(, . \$3\*' &"  
() 3 3>- 3& ( - ' 9' ? > ( 1 , ' ) 9() .

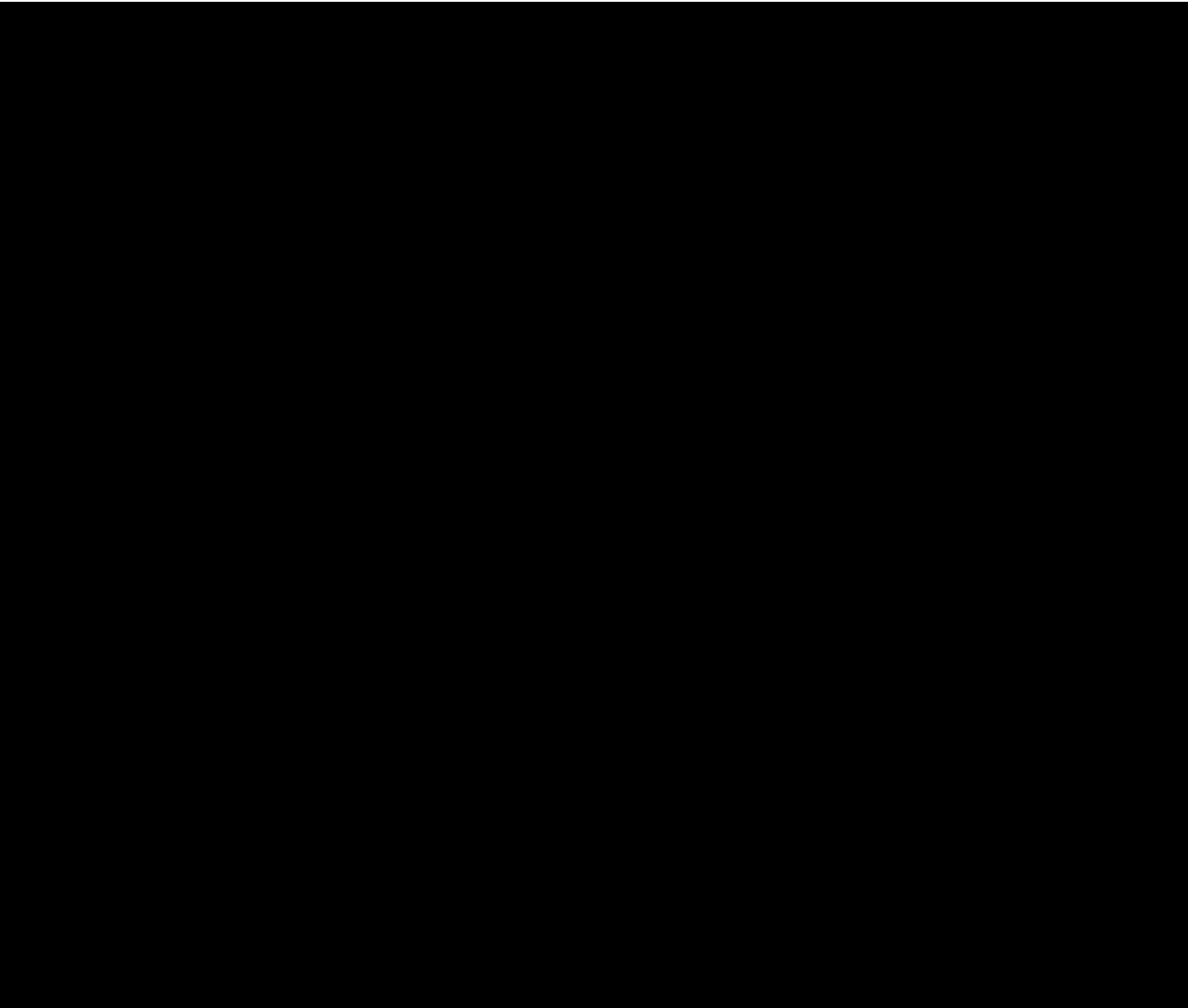
: ' ) 9" 34F%>%- %- 6 + >+ ' \$' 9 - ' @6+ ' %&65 "\$\*" ) 3>' ":

Table 1. An example table

L	R
R	L
L	R

: ' ) 9" +- (\$\*" 3>- 303 \$\*" ) 3>' #" - 5\*3- " @' \*3+ ' - 6 " 3\$\*' 4' \$6+ ' 1\$ @' \*' - %% @' >' - - 68  
) \*3A" , = >+ " ? %- " . .

D\*3A" , = >+ " ? %- " . , %+303 " ) \*' +303 \$\*" ) 3>' 3\$, " ; ' 1 \$#. ) 3 A' @% - ' 180°.



D\*3A", = , ' ) 9" 5' \*' 9\$%\*" @(%\$#. ) ' \*' &%\$\*" ; %938 A(- 9B"%8

$f\left(t, t_{\{c\}}, t_{\{s\}}, \phi_{\{s\}}, \phi_{\{0\}}\right)$

G(- 9B". ) \*%>#\$' +, . %\$ " @ #4. 9(#3; - 3-, " - %8- ( 1 " #+. @6+' %\$ \$ %9(2 ( 1 A' @ ( +- (\$\*" B" 9, 30\*' &&6 # \$\*%4(%&6& (0, 3& , ' ) 9".

H' \$%&' \$"; %93% 3) " #' - "% C\$38 A(- 9B" " &3? - 3 ) 3, (; " \$= + [3].

I%', "@' B". + 93>%) \*" +%>%- ' - " ? %:

```
#pragma once

#include <FixMath.h>

#define LOG(x) String(x) + " " +

typedef float sfix;

const auto tc = (sfix(2*M_PI));
const auto ts (sfix(2*2/3.6*M_PI));
const auto phis (sfix(1.5));
const auto phi0 (sfix(-2));

sfix modc(sfix in, sfix modder)
{
    in = in + modder;
    while(in > modder*sfix(0.5))
    {
        in = in - modder;
    }
    return in;
}

inline sfix Fc(sfix t, sfix phists)
{
    return t*phists;
}

inline sfix Fr0(sfix t, sfix dydx)
{
    return t*dydx;
}

inline sfix Fcomp(sfix ts, sfix phists, sfix dydx)
{
    return Fc(ts*sfix(0.5), phists) - Fr0(ts*sfix(0.5), dydx);
}
```

```

inline sfix Fr(sfix t, sfix ts, sfix phists, sfix dydx)
{
    return Fr0(t, dydx) + Fcomp(ts, phists, dydx);
}

inline sfix Fl(sfix t, sfix ts, sfix phists, sfix dydx)
{
    return -Fr(-t, ts, phists, dydx);
}

inline sfix Ffull(sfix t, sfix tc, sfix ts, sfix phis, sfix phi0)
{
    t = modc(t, tc);
    sfix out;
    auto phists = phis/ts;
    if(t < -ts*sfix(0.5))
    {
        auto dydx = (sfix(2*M_PI) - phis) / (tc - ts);
        out = Fl(t, ts, phists, dydx);
    }
    else if(t < ts*sfix(0.5))
    {
        out = Fc(t, phists);
    }
    else
    {
        auto dydx = (sfix(2*M_PI) - phis) / (tc - ts);
        out = Fr(t, ts, phists, dydx);
    }
    return out + phi0;
}

```

7, . () \*' +, %- ". (0, 3& 9' ? >38 , ') 9" 46, ' " #) 3, =@3+' - ' C9#) %\*" &%- \$', =- '. 4" 4, " 3\$%9'  
**ALFA4C** [2], +>35- 3+, . 1 2' .#. . @693& ) \*30\*' &&" \*3+' - ". \*%' 9\$" +- 65 #" #\$\$& **ALFA** [5].

E#53>- 68 93>) \*" +%>%- - " ?%:

```

#pragma once

#include "RHex.h"
#include "al fa4c.h"
#include "Defines.h"

extern sfix forward_vel;
extern sfix ang_vel;

Channel Sensor<sfix> forward_vel oci ty_command_raw(&forward_vel),
angul ar_vel oci ty_command(&ang_vel);
Channel Computed<sfix> forward_vel oci ty_command([]){ return

```

```

-forward_velocity_command_raw; });
ChannelLast<sfixed> leg_phase, leg_phase_delta;
ChannelLast<sfixed> leg_angles[6], leg_angles_smoothed[6];

MODULE(update_forw_phase,
  È DRIVE(leg_phase, modc(leg_phase.get() + forward_velocity_command.get() *
sfixed(Ts_s), sfixed(2*M_PI)))
)

MODULE(update_delta_phase,
  È DRIVE(leg_phase_delta, modc(leg_phase_delta.get() + angular_velocity_command.get()
* sfixed(Ts_s), sfixed(2*M_PI)))
)

MODULE(stand_up,
  È for(size_t i = 0; i < 6; i++)
  È {
  È // DRIVE(leg_angles[i], phi0 + M_PI * (i%2))
  È DRIVE(leg_angles[i], leg_angles[i] + constrain((phi0+ M_PI * (i%2) -
leg_angles[i]), -2, 2)*Ts_s)
  È }
  È Serial.println("Standup");
  È // DRIVE(leg_angles[1], )
  È // DRIVE(leg_angles[2], )
  È // DRIVE(leg_angles[3], )
  È // DRIVE(leg_angles[4], )
  È // DRIVE(leg_angles[5], )
  È )

MODULE(move_forward,
  È DRIVE(leg_angles[0], modc(Ffull(leg_phase, tc, ts,phis, phi0), sfixed(2*M_PI)))
  È DRIVE(leg_angles[1], modc(Ffull(leg_phase.get() + sfixed(M_PI), tc, ts,phis, phi0),
sfixed(2*M_PI)))
  È DRIVE(leg_angles[2], modc(Ffull(leg_phase, tc, ts,phis, phi0), sfixed(2*M_PI)))
  È DRIVE(leg_angles[3], modc(Ffull(leg_phase.get() + sfixed(M_PI), tc, ts,phis, phi0),
sfixed(2*M_PI)))
  È DRIVE(leg_angles[4], modc(Ffull(leg_phase, tc, ts,phis, phi0), sfixed(2*M_PI)))
  È DRIVE(leg_angles[5], modc(Ffull(leg_phase.get() + sfixed(M_PI), tc, ts,phis, phi0),
sfixed(2*M_PI)))
  È Serial.println("Forward");
  È )

MODULE(rotate_in_place,
  È DRIVE(leg_angles[0], modc(Ffull(leg_phase + leg_phase_delta, tc, ts,phis, phi0),
sfixed(2*M_PI)))
  È DRIVE(leg_angles[1], modc(Ffull(leg_phase + sfixed(M_PI) + leg_phase_delta, tc, ts,
phis, phi0), sfixed(2*M_PI)))
  È DRIVE(leg_angles[2], modc(Ffull(leg_phase + leg_phase_delta, tc, ts,phis, phi0),
sfixed(2*M_PI)))
  È DRIVE(leg_angles[3], modc(Ffull(leg_phase + sfixed(M_PI) - leg_phase_delta, tc, ts,
phis, phi0), sfixed(2*M_PI)))

```

```

Ê   DRIVE(l eg_angles[4], modc(Ffull(l eg_phase - l eg_phase_del ta, tc, ts, phi s, phi 0),
sfi x(2*M_PI)))
Ê   DRIVE(l eg_angles[5], modc(Ffull(l eg_phase + sfi x(M_PI) - l eg_phase_del ta, tc, ts,
phi s, phi 0), sfi x(2*M_PI)))
)

MODULE(move_soft,
Ê   for(size_t i = 0; i < 6; i++)
Ê   {
Ê       // DRIVE(l eg_angles_smoothed[i], l eg_angles_smoothed[i] +
constrain((l eg_angles[i] - l eg_angles_smoothed[i]), -2, 2)*Ts_s)
Ê       DRIVE(l eg_angles_smoothed[i], l eg_angles[i])
Ê   }
)

Updatable *l i nks[] =
{
Ê   &current_real_time_ms,
Ê   &forward_velocity_command_raw,
Ê   &forward_velocity_command,
Ê   &angular_velocity_command,
Ê   &update_forw_phase,
Ê   &update_del ta_phase,
Ê   &l eg_phase,
Ê   &l eg_phase_del ta,
Ê   // &move_forward,
Ê   // &rotate_in_place,
Ê   &stand_up,
Ê   &l eg_angles[0],
Ê   &l eg_angles[1],
Ê   &l eg_angles[2],
Ê   &l eg_angles[3],
Ê   &l eg_angles[4],
Ê   &l eg_angles[5],
Ê   &move_soft,
Ê   &l eg_angles_smoothed[0],
Ê   &l eg_angles_smoothed[1],
Ê   &l eg_angles_smoothed[2],
Ê   &l eg_angles_smoothed[3],
Ê   &l eg_angles_smoothed[4],
Ê   &l eg_angles_smoothed[5],
};

#define LINK_SIZE (sizeof(l i nks)/sizeof(l i nks[0]))

void select_module(Module *module)
{
Ê   l i nks[8] = module;
}

void ALFA_update()

```



```

{
    for(size_t i = 0; i < LINK_SIZE; i++)
    {
        links[i]->update();
    }
}

```

7, . >+"?%- ". +) %\*%> 3#- 3+- ' . ) \*30\*' &&' >3, ? - ' +64\*' \$= #33\$+%\$#\$\$( 1 2 " 8 &3>(, =  
move\_forward " @' ) " #' \$= \$\*%4(%&( 1 #93\*3#\$= + 0, 34', =- ( 1 ) %\*%&%- - ( 1 forward\_vel.

# O1234%&5&

[1] "https://github.com/arsenier/PobeditelRTK"

[2] "https://github.com/arsenier/ALFA4C"

[3] "https://www.desmos.com/calculator/nokfdmo4bt"

[4] [https://www.rhex.web.tr/saranli\\_buehler\\_koditschek.ijrr2001.pdf](https://www.rhex.web.tr/saranli_buehler_koditschek.ijrr2001.pdf) Accessed: Jun. 08, 2024. [Online]. Available: [https://www.rhex.web.tr/saranli\\_buehler\\_koditschek.ijrr2001.pdf](https://www.rhex.web.tr/saranli_buehler_koditschek.ijrr2001.pdf)

[5] E. Gat, "ALFA: a language for programming reactive robotic control systems," in Proceedings. 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA: IEEE Comput. Soc. Press, 1991, pp. 1116-1121. doi: 10.1109/ROBOT.1991.131743.