BULUBOT FW GUIDE v.1.0

8.9.2019

Samet Baykul

FineTuning.cpp

v.1.0.0 : 07.09.2019:

DEFINITION

- Fundamental parameters can be set here

PROPERTIES

SWARM_WIDTH *Static*	*int*	Determines how many robots there are in the swarm
UPDATE_DELAY *Static*	*int*	Determines the time interval between each calculation step in milliseconds
DIAGNOSIS *Static*	*bool*	Turns diagnostic mode on or off. Trace commands do not work when this mode is off
TRACER *Static*	*string*	Expresses the definition of the robot that authorizes printing to the console
TOTAL_MOTOR *Static*	*int*	Determines how many motors are in a Bulubot
TOTAL_DS *Static*	*int*	Determines how many distance sensors are in a Bulubot
TOTAL_LS *Static*	*int*	Determines how many light sensors are in a Bulubot
TOTAL_LED *Static*	*int*	Determines how many LEDs are in a Bulubot
MOTOR_RPM *Static*	*int*	Determines motor speed in rpm
TOTAL_MOTION_TYPES *Static*	*int*	Determines how many motion styles can be in a Bulubot at most
TOTAL_MOTION_FRAME *Static*	*int*	Determines how many frames can be in each motion style at most

Motion.cpp

v.1.0.0 : 07.09.2019:

DEFINITION

- This **class** allows for *creating* and *optimizing* different motions with legs
- Each **motion** consists of multiple **frames**.
- Each **frame** determines which motor is running or not.
- Each **frame** has equal **time intervals**. These time intervals are determined in **steps**.

METHODS

NEW_MOTION ()

Creates a new motion

Parameters	Types	Description
Motion_Index	*int*	The index number which is associated with this motion style
Total_Frame	*int*	The total number of frames which the motion has
Frame_Interval	*int*	Determines the equal time interval which each frame has in steps
Speed	*double*	Determines the speed of motors in radians per second

NEW_FRAME()

Creates a new frame

Parameters	Types	Description
Motor_FL	*bool*	Determines whether the front-left motor is running or not
Motor_FR	*bool*	Determines whether the front-right motor is running or not
Motor_BL	*bool*	Determines whether the back-left motor is running or not
Motor_BR	*bool*	Determines whether the back-right motor is running or not

Output	Types	Description
New_Frame	*FRAME*	This frame is used as a parameter in the "SET_MOTION_FRAME" method

<table>
<tr><td>SET_MOTION_FRAME ()</td><td>This method is used to set a frame inside a motion

<table><tr><th>Parameters</th><th>Types</th><th>Description</th></tr><tr><td>Motion_Index</td><td>int</td><td>The index number which is associated with this frame</td></tr><tr><td>Frame_Index</td><td>int</td><td>The frame index number to be applied on the motion</td></tr><tr><td>Frame</td><td>FRAME</td><td>Frame object instance to be assigned</td></tr></table></td></tr>
<tr><td>SET_AFTER_MOTION ()</td><td>This method is used to assign the desired function to be triggered when a motion is completed.
This method can be used to configure the system control between motions.
The continuity of the motion can also be achieved in this way.

<table><tr><th>Parameters</th><th>Types</th><th>Description</th></tr><tr><td>Callback Function</td><td>Function</td><td>The address of the function to be triggered when the motion is completed</td></tr></table></td></tr>
<tr><td>START ()</td><td>Starts a motion style according to the specified index number

<table><tr><th>Parameters</th><th>Types</th><th>Description</th></tr><tr><td>Motion_Index</td><td>Function</td><td>Index number of the desired motion style</td></tr></table></td></tr>
<tr><td>STOP()</td><td>Stops all current motions</td></tr>
</table>

Framework.cpp

v.1.0.0 : 07.09.2019:

DEFINITION

- All other properties and methods which are ready to use

PROPERTIES

SPEED *Static*	*double*	Speed of the motors [rad/s]
STEP *Static*	*int*	Current calculation step
SELF_DEF *Static*	*string*	Definition of Bulubot instance

FUNDAMENTAL METHODS

INIT() *Static*	This function is triggered once at the start
UPDATE() *Static*	This function is triggered at each step **NOTE:** In order to set execution frequency: Set "Finetuning -> UPDATE_DELAY"

TRACE METHODS

TRACE() *Static*	Print a text

Parameters	Types	Description
Message	*string*	The message which is wanted to be printed

TRACE_INT()
Static

Print an int value with a string label

Parameters	Types	Description
Label	*string*	The label associated with the value
Value	*int*	The value which is wanted to be printed

TRACE_POINT_INT ()
Static

Print an int pair values with string labels

Parameters	Types	Description
Label_1	*string*	The first label associated with the first value
Label_2	*string*	The second label associated with the second value
Value_1	*int*	The first value which is wanted to be printed
Value_2	*int*	The second value which is wanted to be printed

TRACE_DOUBLE ()
Static

Print a double value with a string label

Parameters	Types	Description
Label_1	*string*	The label associated with the value
Value_1	*double*	The value which is wanted to be printed

TRACE_POINT_DOUBLE ()
Static

Print a double pair values with string labels

Parameters	Types	Description
Label_1	*string*	The first label associated with the first value
Label_2	*string*	The second label associated with the second value
Value_1	*double*	The first value which is wanted to be printed
Value_2	*double*	The second value which is wanted to be printed

TRACE_END()
Static

You have to call this function when you are sure to finish the print line. If not, you will GET AN ERROR

SENSOR METHODS

READ_SENSOR_DS_L()
Static

Returns value of lefts distance sensor

Output	Types	Description
Value	*double*	Measured current sensor data

READ_SENSOR_DS_R()
Static

Returns value of right distance sensor

Output	Types	Description
Value	*double*	Measured current sensor data

READ_SENSOR_LS ()
Static

Returns value of light sensor

Output	Types	Description
Value	*double*	Measured current sensor data

MOTION METHODS

SET_MOTOR_FL_SPEED ()
Static

Set speed of the front-left motor

Parameters	Types	Description
Speed	*double*	Speed in radians per second

SET_MOTOR_FR_SPEED ()
Static

Set speed of the front-right motor

Parameters	Types	Description
Speed	*double*	Speed in radians per second

SET_MOTOR_BL_SPEED ()
Static

Set speed of the back-left motor

Parameters	Types	Description
Speed	*double*	Speed in radians per second

SET_MOTOR_BR_SPEED ()
Static

Set speed of the back-right motor

Parameters	Types	Description
Speed	*double*	Speed in radians per second

SUPERVISOR METHODS

GET_POSITION_X ()
Static

Returns the position of the given robot on the x axis

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Position_X	*double*	The position of the given robot on the x axis

GET_POSITION_Z ()
Static

Returns the position of the given robot on the z axis

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Position_Z	*double*	The position of the given robot on the z axis

GET_ROTATION_DEGREE ()
Static

Returns the rotation angle of the given robot in 2D as degree format

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Rotation_Angle	*double*	The rotation angle of the given robot in degree

GET_ROTATION_RADIAN ()
Static

Returns the rotation angle of the given robot in 2D as radian format

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Rotation_Angle	*double*	The rotation angle of the given robot in radian

GET_SPEED_X ()
Static

Returns the speed of the given robot on the x axis

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Speed_X	*double*	The speed of the given robot on the x axis

GET_SPEED_Z ()
Static

Returns the speed of the given robot on the z axis

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Speed_X	*double*	The speed of the given robot on the z axis

GET_SPEED ()
Static

Returns the speed of the given robot

Parameters	Types	Description
Robot_Index	*int*	Index number of the target robot in “LIST_ROBOTS” array

Output	**Types**	**Description**
Speed	*double*	The speed of the given robot

EXAMPLE

```
// ------------------------------------------------
// Bulubot Controller Manin
// v.1.0.0
// by Samet Baykul
// ------------------------------------------------

// ------------------------------------------------
// SUPERVISOR
// ------------------------------------------------

// Input all robot definitons in thw swarm here

char LIST_ROBOTS[SWARM_WIDTH][8] = {"BB_1", "BB_2"};

//-------------------------------------------------
// DEFINITIONS
//-------------------------------------------------

// Define your functions here

void Set_Motion();
void Motion_Update();

//-------------------------------------------------
// MAIN
//-------------------------------------------------

void INIT()
{
  // This function is triggered once at the start
  // Motion Creation

  Set_Motion();
  MOTION.START(0);
}

void UPDATE()
{
  // This function is triggered at each step
  // In order to set execution frequency: Set "FineTunning -> UPDATE_DELAY"
  // These are example of using some important methods which includes some TRACE and SUPERVISOR methods

  TRACE(SELF_DEF);
  TRACE_POINT_DOUBLE("BB_1 || POS_X", "POS_Z", GET_POSITION_X(0), GET_POSITION_Z(0));
  TRACE_DOUBLE("ROT", GET_ROTATION_DEGREE(0));
  TRACE_DOUBLE("SPEED", GET_SPEED(0));
  TRACE_END();
  TRACE_POINT_DOUBLE("BB_2 || POS_X", "POS_Z", GET_POSITION_X(1), GET_POSITION_Z(1));
  TRACE_DOUBLE("ROT", GET_ROTATION_DEGREE(1));
  TRACE_DOUBLE("SPEED", GET_SPEED(1));
  TRACE_END();
```

```
}

//-------------------------------------------------
// FUNCTIONS
//-------------------------------------------------

void Set_Motion()
{
  // Example for creation of a new motion
  // You can edit or define new motion like this

  MOTION.NEW_MOTION(0, 9, 16, SPEED);
  MOTION.SET_MOTION_FRAME(0, 0, MOTION.NEW_FRAME(0, 0, 1, 0));
  MOTION.SET_MOTION_FRAME(0, 1, MOTION.NEW_FRAME(1, 0, 1, 0));
  MOTION.SET_MOTION_FRAME(0, 2, MOTION.NEW_FRAME(1, 0, 1, 0));
  MOTION.SET_MOTION_FRAME(0, 3, MOTION.NEW_FRAME(1, 0, 0, 0));
  MOTION.SET_MOTION_FRAME(0, 4, MOTION.NEW_FRAME(0, 0, 0, 1));
  MOTION.SET_MOTION_FRAME(0, 5, MOTION.NEW_FRAME(0, 1, 0, 1));
  MOTION.SET_MOTION_FRAME(0, 6, MOTION.NEW_FRAME(0, 1, 0, 1));
  MOTION.SET_MOTION_FRAME(0, 7, MOTION.NEW_FRAME(0, 1, 0, 0));
  MOTION.SET_MOTION_FRAME(0, 8, MOTION.NEW_FRAME(1, 1, 1, 1));
  MOTION.SET_AFTER_MOTION(Motion_Update);
}

void Motion_Update()
{
  // This is a call-back function. It is triggered whenever a motion is completed

  MOTION.START(0);
}
```

ERROR CODES			
Error Code	Related Method	Possible Sources	Suggestions
001	Motion.cpp NEW_MOTION() SET_MOTION_FRAME()	**Motion_Index Error:** The parameter 'Motion_Index' should be less than FineTuning parameter 'TOTAL_MOTION_TYPES'	Increase the parameter, FineTuning.cpp -> TOTAL_MOTION_TYPES
002	Motion.cpp NEW_MOTION() SET_MOTION_FRAME()	**Total_Frame Error:** The parameter 'Frame_Index' should be less than FineTuning parameter 'TOTAL_MOTION_FRAME'	Increase the parameter, FineTuning.cpp -> TOTAL_MOTION_FRAME