فهرست:

تقدیم

The All

Power By : Sarbaz13
Software : Webosts
Weblog: www.egs13.blogfa.com
DOn'a Copy Right

چکیده:

این پروژه در چهار فصل گرد آوری شده است که فصل اول آن آشنایی با Webots است که تاریخچه و کاربردهای این نرم افزار و مزیت های آن توضیح داده شده است و فصل دوم آن شروع کار با webots است که در این فصل انواع منوها و دکمه هایی که در این نرم افزار تعبیه شده اند توضیح داده است فصل سوم این پروژه برنامه نویسی و کنترل است که در آن طرح کلی برنامه از دیدگاه webots و انتقال برنامه به ربات واقعی و کنترل از راه دور توضیح داده شده است و بالاخره در فصل چهارم این پروژه انواع گره ها و دستوراتی را که در این نرم افزار استفاده می شود توضیح داده می شود.

فصل اول

آشنایی با Webots

۱-۱-مقدمه

وبتز نرم افزاری برای مدلسازی و شبیه سازی سریع رباتهای متحرک است که از سال ۱۹۹۶ توسعه یافته و توسط دکتر اولیور میشل عضو انجمن تکنولوژی فدرال سویس در آزمایشگاه جین دانیل نیکود طراحی شد. از سال ۱۹۹۸ Webots یک محصول تجاری شد و بوسیله شرکت سایبربوتیک توسعه یافت.

کاربرانی که آن را خریدند بالغ بر ۴۰۰ دانشگاه و مرکز تحقیقاتی در سراسر جهان هستند. وبتز همچنین توسط ارگانهای محقق و مراکز تحقیقاتی شامل تویوتا هوندا سونی پاناسونیک پایونیر ، ن تی تی ، سامسونگ ، ناسا ، استنفورد و غیره استفاده می شود.

Webots یک بسته نرم افزاری شبیه سازی رباتهای متحرک به صورت حرفه ای است که یک مدل سریع همره با محیط آن از یک ربات را ارائه می کند.

وبتز به کاربران اجازه می دهد که یک جهان مجازی سه بعدی همراه با خصوصیات فیزیکی شبیه جرم ، مفاصل ، ضریب اصطکاک و غیره را خلق کنند. کاربر به راحتی می تواند اشیاء غیر فعال و فعال خود را که رباتهای متحرک باشند را اضافه کند.

این رباتها می توانند مکانیزم های جابه جایی متفاوت داشته باشند نظیر رباتهای چرخ دار ، پادار ، رباتهای پرنده و زیر آبی . علاوه بر این ، آنها را می توان به بسیاری از سنسورها و موتورهای محرک مجهز نمود ، مانند سنسورهای فاصله یاب ، چرخهای دیفرانسیلی ، دوربین ، سرو موتور ، سنسورهای تماسی ، دست گیرنده ، ارسال کننده ، دریافت کننده و غیره.

در پایان کار ، کاربر می تواند هر ربات را به صورت جداگانه برنامه نویسی کند تا رفتار منحصر به فرد خود را داشته باشد. همچنین وبتز دارای مثالها و مدلهای مختلفی از انواع رباتها و برنامه کنترلی آنها می باشد که از طریق پوشه پروژه و help آن قابل دسترسی می باشند.

وبتز دارای تعدادی رابط به رباتهای متحرک واقعی نیز می باشد که به طوری که شما با شبیه سازی و برنامه نویسی رباتتان ، قادر خواهید بود که آن را به ربات واقعی منتقل کنید. مانند رباتهای

e-puck , Khepera , Hemisson , LEGO Mindstorms , Aibo

استفاده از شبیه سازی سریع برای توسعه پروژه های رباتیک بسیار مفید است و عملا به طراحان اجازه می دهد تا سریعا ایده خود را ببینند و داده ها را به ربات واقعی انتقال دهند.

استفاده از چنین نرم افزاری وقت بسیاری را ذخیره می کند و به طراح این امکان را می دهد تا احتمالات بیشتری را بسنجد . بنابراین کنترل زمان و کیفیت ، نتیجه استفاده از این نرم افزار است.

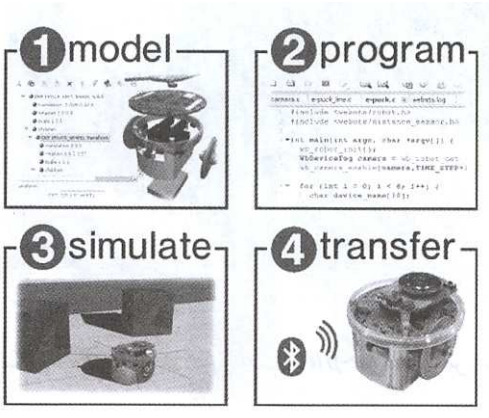
۱-۳- قابلیتهای Webots

وبتز برای کارهای تحقیقاتی و پروژه های آموزشی و دانشگاهی در ارتباط با رباتهای متحرک بسیار مناسب است . وبتز در این زمینه ها می تواند استفاده شود :

۱. مدل کردن رباتهای متحرک شامل تحقیقات آکادمیک ، اتوماسیون صنعتی ، هوانوردی ، صنعت جارو برقی ، صنایع اسباب بازی و سر گرمی ها
۲. تحقیقات در زمینه مکانیزمهای خود جابه جایی رباتها (پا دار ، انسان نما ، چهار پا و چرخ دار)
۳. تحقیقات رباتهای گروهی مانند هوش جمعی و همکاری گروه های رباتی .
۴. تحقیقات رفتارهای انطباقی : الگوریتمهای ژنتیک ، شبکه های عصبی ، یادگیری انطباقی و هوش مصنوعی.

۱-۴- مراحل طراحی در وبتز

همانطور که در شکل ۳ دیده می شود طراحی یک ربات در Webots از چهار مرحله تشکیل شده است.

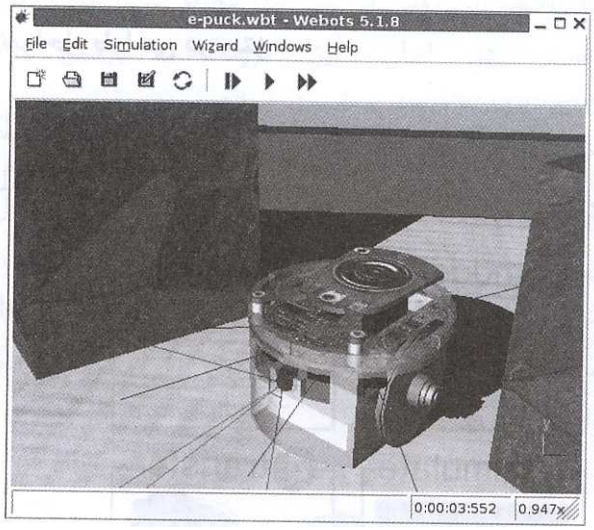


۱. **مدل کردن (Modeling):**

ساختن شکل فیزیکی ربات شامل ، اضافه کردن موتورها ، سنسورها ، بدنه ربات و ساختن محیط ربات ؛ در این مرحله می توان هندسه اشکال و رنگ آنها را تغییر داد.

و همچنین خصوصیات فنی سنسورها را تعریف کرد و از این طریق هر نوع رباتی را ساخت ، اعم از رباتهای چرخ دارد ، رباتهای پادار ، رباتهای انسان نما ، پرنده ، رباتهای زیر آبی و بازوهای مکانیکی ، همچنین محیط ربات را به طریق مشابه شامل موانع ، دیوارها ، توپ و سایر عوارض مصنوعی و طبیعی مدل کرد.

تمامی خصوصیات اشیاء مانند جرم ، ممان اینرسی ، ضریب اصطکاک ، ضریب جهش و غیره قابل تعریف هستند . هنگامی که محیط و ربات ساخته شده ، کاربر می تواند به مرحله بعدی طراحی برود.



۲. **برنامه نویس (Programming)**:

بر اساس کاری که باید انجام شود برای جلوگیری از برخورد و رسیدن به هدف ؛ شامل برنامه نویسی گرافیکی برای مبتدیان تا سطح پیشرفته که به زبانهای Matlab , c,c++, java , Phyton می توان به دلخواه برنامه نویسی انجام داد. برنامه کنترلی ربات به طور کلی یک حلقه بی انتهاست که شامل سه قسمت است :

۱- خواند مقادیر اندازه گیری شده توسط سنسورها.
۲- محاسبه آن چیزی که باید حمل بعدی ربات باشد.
۳- فرستادن فرمان به محرکها و موتورها برای انجام این عمل آسانترین قسمت های ، قسمت ۱ و ۳ است و قسمت ۲ مشکل ترین قسمت برنامه است که هوش مصنوعی در این قسمت خود را نشان خواهد داد.

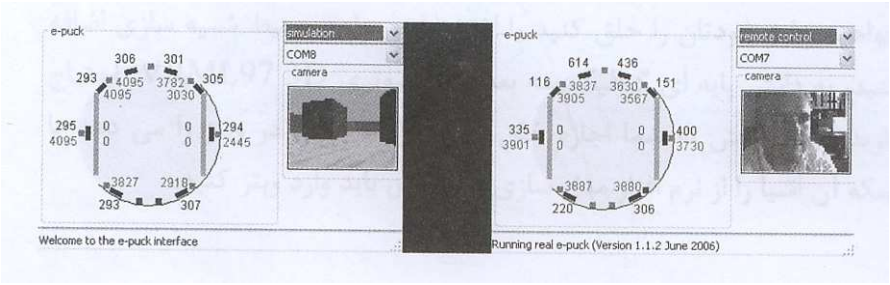
۳. **شبیه سازی حرکت (Simulation)**:

یعنی حرکت دادن ربات در محیط آن به وسیله برنامه همراه آن به منظور دیدن نتایج مطلوب و اصلاح آنها ، این قسمت به کاربر اجازه خواهد داد که رفتار برنامه کنترلی را تست کند.

در این قسمت با حرکت دادن موانع و تغییر محیط بوسیله موس می توان ربات را با شرایط محیطی مختلف به حرکت واداشت و نهایتا بهترین ربات را برای ساخت انتخاب کرد.

۴. **انتقال برنامه :**

(Transfer) به ربات واقعی بوسیله پورت کامپیوتری یا ارتباط بی سیم ؛ در این قسمت برنامه کنترلی ربات به ربات واقعی انتقال داده می شود.



اگر ربات شما در این مرحله به خوبی محیط شبیه سازی عمل کرد که به نتیجه دلخواه رسیده اید در غیر این صورت دوباره به مرحله یک رفته و شبیه سازی و برنامه را اصلاح و بهتر می کنید و دوباره انتقال می دهید. ولی در اکثر مواقع با یک تغییر کوچک به نتیجه دلخواهتان خواهید رسید.

فصل دوم

شروع کار با : Webots

اگر چه هیچ دانش خاصی برای دیدن مدلهای شبیه سازی شده وبتز احتیاج نیست . اما به مقدار کمی دانش فنی برای تکمیل شبیه سازی احتیاج است.

دانستن پایه ای یکی از زبانهای Phyton ، c,c++, java برای نوشتن برنامه کنترل ربات مورد نیاز است . با این حال حتی اگر این زبانها را ندانید باز هم می توانید رباتهای Hemisson , e-puck را بوسیله زبان برنامه نویسی گرافیکی موسوم به Botstudio برنامه نویسی کنید.

اگر شما نمی خواهید از مدلهای تهیه شده در وبتز استفاده کنید و می خواهید ربات خودتان را خلق کنید یا اشیا خاصی را به محیط شبیه سازی اضافه کنید ، به دانش پایه ای گرافیک سه بعدی کامپیوتری VRML 97 3D احتیاج دارید و این دانش به شما اجازه خلق یک مدل ۳ بعدی در وبتز را می دهد یا اینکه آن اشیا را از نرم افزار مدل سازی سه بعدی باید وارد وبتز کنید.

۲-۱- شبیه سازی در وبتز

شبیه سازی در وبتز از اجزای زیر تشکیل می شود:

۱. فایل Webots world که معرف یک یا چند ربات سه بعدی و محیط آنها است.
۲. برنامه های کنترلی برای رباتهای بالا
۳. یک ناضر (سوپروایزر) اختیاری

۲-۱-۱- جهان مجازی چیست؟ World

یک جهان در وبتز توصیف سه بعدی از خصوصیات رباتها و محیط آنها است. شامل توصیف همه چیز ، موقعیت ربات ، جهت گیری ، هندسه ، ظاهر (مانند رنگ و نور) ، خصوصیات فیزیکی ، انواع اشیا و غیره . جهان مجازی به صورت سلسله مراتبی (درختی) سازمان می یابد.

بطوری که هر شی می تواند اشیا دیگر را در بر داشته باشد. برای مثال یک ربات می تواند شامل دو چرخ ، یک سنسور فاصله یاب و سرو موتور باشد که خود سرو شامل یک دوربین باشد.

فایل World شامل کدهای کنترلی نیست بلکه فقط نام برنامه کنترلی مورد نیاز هر ربات را مشخص می کند. فایل World با پسوند wbt . ذخیره می شود که این فایلهای wbt. در پوشه Worlds هر ربات ذخیره می شوند.

۲-۱-۲- کنترلر یا برنامه کنترلی چیست؟

کنترلر یک برنامه کامپیوتری است که یک ربات خاص که با فایل World آن مشخص شده است را کنترل می کند. کنترل می تواند ، با هر یک از زبانهای برنامه نویسی پشتیبانی شده توسط وبتز نوشته شود. شامل:

C,c++ , java , Matlab , python , URBI هنگامی که شبیه سازی به حرکت درآمد ، وبتز یک کنترلر خاص را آغاز می کند که به عنوان یک فرآیند مجزا عمل می کند و در ارتباط با رباتهای شبیه سازی شده است. توجه داشته باشید که چندین ربات می توانند از یک برنامه کنترلی یکسان استفاده کنند ولی فرآیند ی مجزا برای هر کدام آغاز می شود.

بعضی از زبانهای برنامه نویسی نیاز به مفسر برنامه دارند (c,c++) که در خود وبتز تفسیر و خطایابی می شوند. بعضی دیگر نیاز به ترجمه دارند مانند (Python ,URBI) و بعضی دیگر مانند java نیاز به هر دوی ترجمه و تفسیر دارند (در وبتز انجام می شود).

فایلهای binary , source هر کنترلر در پوشه کنترل آن ذخیره می شود. پوشه کنترلر در زیر پوشه اصلی پروژه هر ربات قرار دارد که توسط وبتز ساخته خواهد شد و شرح آن بعدا گرفته خواهد شد.

۲-۱-۳- ناظر (سوپروایزور) چیست؟

ناظر یک ربات ویژه است که کارهایی را انجام می دهد که بطور طبیعی فقط توسط انسان انجام می شود و نه توسط یک ربات واقعی ، ناظر با برنامه کنترلی در ارتباط است که به هر کدام از زبانهای توضیح داده شده می تواند نوشته شود.

اما در تقابل با کنترلر رباتها یک ناظر می تواند به اعمال ویژه ای دست یابد. این اعمال شامل کنترل شبیه سازی است برای مثال ، حرکت رباتها به یک موقعیت تصادفی ، گرفتن فیلم از محیط شبیه سازی و غیره .

در واقع ناظر مانند انسانی است که در کنار رباتها ایستاده و آنها را کمک و بررسی می کند برای مثال در مسابقات رباتهای فوتبالی آن فردی که به عنوان مربی و سازنده رباتهاست و در کنار زمین ایستاده است به عنوان ناظر است و می تواند دکمه شروع ربات و موقعیت آن را تغییر دهد و از زمین بازی خارجش کند و کارهای دیگر ، ولی نمی تواند در کار اصلی ربات دخالت تا به جا کند و نقش ربات را کمرنگ کند.

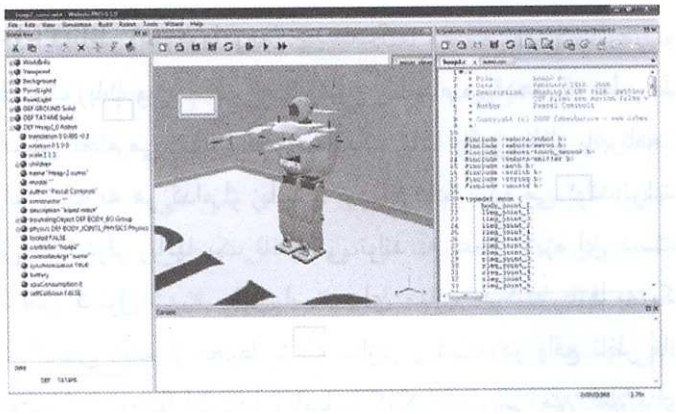
۲-۲- محیط اجرایی نرم افزار

برای شروع نرم افزار از منوی استارت کامپیوتر مسیر زیر را طی می کنید.

Start / program files / cyberbotics / webotes 6

یا از میانبر دسکتاب کامپیوتر آن را آغاز کنید. پس آغاز برنامه لیست گزینه هایی که می توان وبتز را با آن شروع کرد ظاهر می شود که یکی را باید انتخاب نمود. محیط اجرایی نرم افزار از چهار جز تشکیل شده که هر کدام یک قسمت از وظایف را بر عهده دارند و در چهار پنجره جداگانه هستند .

مانند شکل ۲-۱



۱. پنجره نمایش شبیه سازی (Simulation):

که شکل نهایی ربات و محیط آن را نشان می دهد و زمان سپری شده را نیز در زیر آن نشان می دهد و می توان برنامه را از این پنجره اجرا یا متوقف کرد.

۲. پنجره درختی Scene tree :

اصلی ترین پنجره نرم افزار است و وسایل ساخت ربات در این قسمت وجود دارد که بصورت درختی و زیر مجموعه ای تمام اجزاری موجود در پنجره نمایش دارای اسم ، مرز و فیزیک هستند .

هر عنصری که زیر مجموعه دیگری باشد بوسیله بالاتر یا در اصطلاح والدین گردانیده می شود. والدین ، از بچه ها یا children تشکیل شده است و بچه ها توسط والدین هدایت می شوند. فرضا یک سرو موتور از یک چرخ و یک محور درست شده است.

چرخ و محور بچه های سرو هستند و سرو ، چرخ و محور را می گرداند و خود چرخ و محور به طور مجزا دارای یک مرز مشخص و یک فیزیک مشخص شامل جرم و خصوصیات جرمی هستند.

۳. پنجره برنامه نویسی (text editor window):

هر ربات دارای حداقل یک برنامه کنترلی می باشد که از طریق این پنجره که یک کامپایلر یا مفسر زبان های c و c++ و جاوا است ، نوشته می شود.

این پنجره خطاهای برنامه را پیدا می کند و فایل های لازم را بوسیله دکمه built می سازد که در پوشه کنترل زیر مجموعه پوشه ربات قرار می گیرد.

گاهی لازم است برای محیط هم ویژگی های خاصی تعریف شود که در همین پنجره بوسیله برنامه نویسی و استفاده از موتور دینامیکی باز یا ode قابل تعریف است . این موتور دارای یک سری از دستورات سطح بالای زبان c برای دسترسی به اجزا ربات است . مانند این دستور:

Void dBody AddForce (dBody ID, dReal fx,dReal fy , dReal fz);

که نیرویی در سه جهت به یک جسم با شناسه مشخص اضافه می کند و درباره آن در فصل های بعدی شرح بیشتری داده خواهد شد.

۴. پنجره گزارش (console window):

که هر گونه نتیجه ای مانند سرعت ، موقعیت یا نیرو ربات در آن به صورت لحظه ای قابل چاپ و نمایش است. این پنجره برای دانستن لحظه به لحظه اطلاعات ربات بسیار مفید است. در واقع آن متغیرها یا پیامهایی که در برنامه کنترلی بوسیله دستور پرینت صادر می شود در این پنجره نمایش داده می شود.

۲-۲-۱- پنجره اصلی : منوها و دکمه ها

پنجره اصلی دارای هشت منو است شامل :

File , Edit , View , Simulation , Build , Tools , Wizard , Help

File menu که انجام کارهای معمول بر عهده آن است مانند ذخیره کردن و دارای گزینه های زیر است:

New world

(دکمه و ایتم) یک جهان مجازی را در پنجره شبیه سازی همراه با یک زمین مربعی با ده ضربدر ده مربع و یک متر در یک متر مساحت و یک چراغ ایجاد می کند که زمین آن گره ElevationGrid می باشد.

Open World

(دکمه و ایتم) یک پنجره انتخاب را باز می کند که در آن می توان یک فایل اجرایی wbt را اجرا کرد.

Save world

(دکمه و ایتم) جهان مجازی را که اکنون وجود دارد را با نامش ذخیره می کند. نام فایل در بالای پنجره اصلی نشان داده می شود. در هر بار ذخیره کردن فایل wbt . بازنویسی می شود و هیچ کپی پشتیبانی بوسیله وبتز از حالت قبلی بوجود نمی آید. بنابراین زمانی باید از این دکمه استفاده نمود که از تغییرات مطمئن بود.

Save World As...

(دکمه و آیتم) جهان مجازی فعلی را با یک نام جدید که بوسیله کاربر وارد می شود ذخیره می کند . توجه کنید که فایل wbt . باید همیشه در پوشه پروژه وبتز (Project) و در زیر پوشه Worlds ذخیره شود در غیر اینصورت قادر به باز کردن مجدد آن فایل نخواهید بود.

Revert World

(دکمه و آیتم) جهان مجازی و شبیه سازی را دوباره از آنجایی که ذخیره شده دوباره بار گذاری و اجرا می کند و شبیه سازی را از نو شروع می کند.

New Text File

(دکمه و آیتم) یک صفحه ویرایش متن جهت کد نویسی را در پنجره ویرایشگر باز می کند.

Open Text File

(دکمه و ایتم) یک پنجره انتخاب باز می کند که به شما اجازه یک فایل متنی را می دهد (مانند Java file)

Save Text File

(دکه و آیتم) فایل متنی جاری را ذخیره می کند.

Save Text File As...

(دکمه و آیتم) اجازه ذخیره کدها را با یک نام و پسوند جدید در پوشه کنترل زیر مجموعه پوشه اصلی ربات می دهد.

Revert Text File

(دکمه و آیتم) فایل متنی را از آنجایی که ذخیره شده است را دوباره به نمایش در می آورد.

- **Page setup**

(ایتم) پنجره ای را گشوده و اجازه مدیریت صفحه به منظور چاپ فایل متنی را می دهد.

- **Print**

(آیتم) پنجره چاپ محتویات فایل متنی کنترل برنامه را باز می کند.

- **Import VRML 2.0**

یک شی مدل شده از سایر نرم افزارهای مدلسازی را که با استاندارد VRML 97 ساخته شده است را در انتهای گره ها وارد کرده و در پنجره شبیه سازی نمایش می دهد. این فایل با پسوند wrl باید ذخیره شده باشد.

برنامه هایی که چنین خروجی دارند مانند 3D Studio , Solidwroks , Max , Maya Auto CAD, Pro Engineer , AC3D می باشند. توجه داشته باشدی که وبتز vrml 1 را پشتیبانی نمی کند.

هنگامی که این شی وارد شد به عنوان یک گره Group , Transform یا shape شناخته می شود و در انتهای لیست گره ها قرار می گیرد و بعد از آن کاربر می تواند آنها را هر جا که تمایل دارد قرار دهد ، مثلا زیر مجموعه یک جسم صلب.

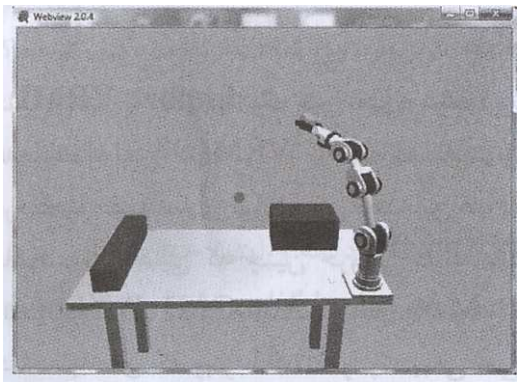
- **Make Animation** :

این آیتم یک انیمشن سه بعدی با پسوند wva از محیط شبیه سازی درست می کند. این فرمت را می توان در نرم افزار رایگان Web view نمایش داد که از سایت شرکت سایبربوتیک قابل دانلود است.

این نرم افزار اجازه دیدن انیمیشن از هر زاویه ای را میسر می سازد و قابل ملحق شدن به مرور گرهای اینترنت نیز می باشد. شکل ۲-۲ .

Web view نرم افزار مناسبی برای دیدن نتایج وبتز ، مخصوصا در اینترنت است. این نرم افزار بصورت مستقل نیز می تواند انیمیشنهای وبتز را نشان دهد و پس از نصب ایکون آن در دسکتاب ظاهر می شود.

شکل ۲-۲



- **Make Movie**:

این گزینه یک فیلم با فرمت MPEG یا AVI تولید می کند. توجه کنید فیلمبرداری از زمانی که شبیه سازی به حرکت و داشته می شود شروع می می شود و تا زدن دکمه توقف ادامه می یابد آنگاه پنجره ذخیره کردن باز می شود.

اما این فیلمها با سرعت ۲۵ عکس بر ثانیه هستند به همین علت سرعت پخش فیلم ممکن است سریعتر یا آهسته تر از سرعت شبیه سازی و سرعت واقعی باشد.

- **Take Screen shot**:

این گزینه یک عکس از صفحه جاری محیط شبیه سازی می گیرد و با فرمت PNG قابل ذخیره است.

- **Quit Webots** :

باعث تمام شدن شبیه سازی و بسته شدن پنجره وبتز می شود.

۲-۲-۲- منوی Edit

Undo: برای بازگردانی آخرین تغییرات در ویرایشگر متن کدها است.

Redo : برای باز گرداندن گزینه undo است.

Cut, Copy, Paste, Select all : این آیتم ها مانند سایر نرم افزارها عمل می کنند.

Find : برای جستجوی یک متن خاص در ویرایشگر متن است.

Replace : برای جستجوی و جاگذاری متن موجود با متن جدید است.

Go to line : برای رفتن نشانگر به یک خط خاص است.

۲-۲-۳- منوی View

این منو کنترل نما و زاویه دید پنجره شبیه سازی را انجام می دهد. شامل این گزینه ها می باشد.

- **Follow Object :**

این گزینه که با کلید F5 نیز فعال می شود. کار تعویض بین نمای ثابت و نمای پویا را بر عهده دارد. در نمای پویا تصویر یک شی معمولا ربات را دنبال می کند و نیاز به تغییر نما دیگر نیست.

اگر شما نیازمندید که نقطه دید خود را عوض کنید نخست باید آن شی را با موس انتخواب کرده سپس گزینه Follow Object را انتخاب کنید و این تغییر با ذخیره کردن فایل نیز ذخیره می شود.

- **Restore Viewpoint :**

این گزینه نقطه و زاویه دید را به حالت اولیه اش بر می گرداند یعنی به آخرین زمانی که ذخیره شده است. این گزینه همچنین حالت Projection را به Perspective تغییر می دهد. این گزینه زمانی به کار می رود که شما زاویه دید صفحه را گم کرده باشید با این گزینه به نمای اولیه بر می گردد.

- **Projection :**

زیر منوی این گزینه شامل نماهای Perspective(سه نما) و Orthographic است که مد سه نما اولویت برنامه است . این مد تصویر طبیعی را می دهد که شی را کوچکتر نشان می دهد ولی نمای دوم ، فاصله از شی ، در بزرگی آن تاثیری ندارد. علاوه بر این در مد ارتوگرافیک خطهای موازی با تصویر هستند.

- **Rendering :**

دارای دو زیر منوی Regular و Bounding Objects است. در حالت اول که اولویت هم برنامه است ، شی دارای ظاهری هندسی ، رنگی و توپر است و با چشم و دوربین به خوبی دیده می شود.

در مد Bounding Objects تنها مرز اشیا که به صورت خطهای سفید رنگ است نشان داده می شود. این حالت برای حل مساله کشف بر خورد مناسب است.

- **Show Contact Points :**

این گزینه نقاط تماس ربات با محیط را نشان می دهد و موتور کشف بر خورد را روشن می کند. توجه کنید که نقاط تماسی که در آنجا یک نیروی تماسی ایجاد نشده باشد نشان داده نمی شوند.

یک نیروی تماسی تنها برای اشیایی بوجود می آید که دارای فیزیک (جرم) باشند (گره فیزیک آنها خالی نباشد).

۲-۲-۴- منوی Simulation

برای کنترل اجرای شبیه سازی استفاده می شود شامل گزینه های زیر است :

Stop : (دکمه و آیتم) شبیه سازی را متوقف می کند.

Step : (دکمه و آیتم) یک گام از شبیه سازی را اجرا می کند. مدت زمان این گام در فیلد basic Time Step زیر مجموعه گره World info مشخص می شود.

Run : (دکمه و آیتم): شبیه سازی را اجرا می کند تا هنگامی که یکی از دکمه های stop یا step کلیک شود. در حالت شبیه سازی ، صفحه نمایش هر n گام زمانی یکبار تازه می شود. که n نیز در فیلد display Refresh زیر مجموعه گره World info تعریف می شود.

Fast : (دکمه و آیتم) شبیه به دکمه Run است با این تفاوت که نمایش گرافیکی دیگر وجود ندارد. هنگامی که صفحه نمایش غیر فعال می شود (صفحه سیاه) این کار باعث می شود تا شبیه سازی سریعتر انجام شود(فقط در ویرایش حرفه ای وبتز فعال است Webots PRO) و برای کارهایی با پردازش زیاد مفید است . مانند ژنتیک الگوریتم ، پردازش تصویری ، یادگیری و غیره.

۲-۲-۵- منوی Build

Compile : ترجمه زبان برنامه نویسی و خطایابی کدها و دستورات نوشته شد در ویرایشگر را با کمک قابل Make file که باید در پوشه کنترل ربات به صورت دستی یا بوسیله گزینه new controller در منوی ویزارد کپی شده باشد ، انجام می دهد.

توجه کنید این فایل را می توان از سایر پوشه های کنترلی واقع در رباتهای مثال وبتز نیز کپی کنید. این دکمه فقط زبانهای java , c++, c را ترجمه یا کامپایل می کند.

Build : این دکمه فایلهای اجرایی لازم برای کنترل شبیه سازی را با کمک make file می سازد و در پوشه کنترل ربات قرار می دهد. این دکمه با آیتم بعد از ترجمه و خطایابی برنامه باید اجرا شود تا خطایی اتفاق نیفتد. زدن این دکمه و ساختن فایلهای اجرایی الزاما برای کنترل ربات باید صورت گیرد.

Clean : این گزینه فایلهای اجرایی ساخته شده در پوشه کنترل توسط گزینه build را حذف خواهد کرد.

- **Make JAR file :** این گزینه اجازه ساختن یک فایل jar را از java رابه کاربر می دهد.

Cross – Compile : این آیتم اجازه Cross – Compile (انتقال کدها به پردازش گر ربات واقعی) را از دستورات جاری نوشته شده در ویرایشگر به کاربر می دهد. توجه کنید برای این کار به یک Make file ویژه احتیاج است مثلا برای ربات e-Puck نام آن باید چنین باشد e-Puck . Make file.

Cross – compilation clean : این دکمه یا گزینه فایلهای cross – compilation را حذف خواهد کرد. برای این کار نیز مانند آیتم قبل باید Make file ویژه در پوشه کنترل ربات کپی شده باشد.

۲-۲-۶- منوی Robot

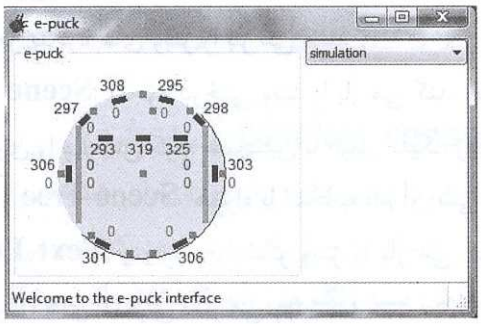
- **Control Edit :**

پنجره ویرایشگر کدها را باز می کند.

- **Robot Window :**

پنجره رباتی را باز می کند که نوع این پنجره وابسته به نوع ربات است. (شکل ۲-۳) . بعضی رباتهای ممکن است این پنجره را نداشته باشند. وبتز پنجره های ویژه ای برای رباتهای e – puck , Khepera Aibo دارد.

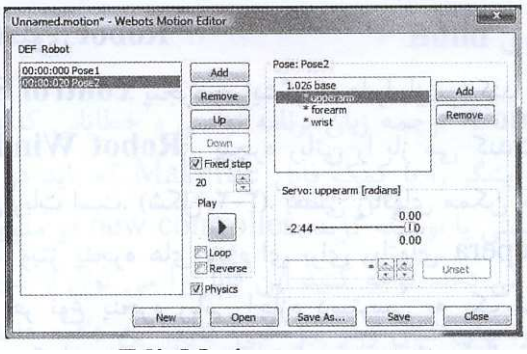
هر نوع پنجره رباتی اجازه دسترسی به یک سطح از ارتباط با سنسورها و محرکهای ربات را می دهد. این منو زمانی فعال می شود که ربات انتخاب شده باشد.



: Editor Motion

پنجره ای با همین نام را باز می کند که حرکات ربات را ویرایش می کند (شکل ۲-۴) . این پنجره وسیله ای برای طراحی حرکات متوالی رباتهای مفصل دار است یعنی نخست یک مفصل بچرخد سپس دیگری و ... این ترتیب ها در یک فایل به نام motion files ذخیره می شود و سپس در کدهای کنترلی محسوب می شود.

توجه داشته باشید این پنجره فقط برای رباتهایی که دارای سرو موتور هستند فعال می شود. درباره این پنجره در بخشهای بعد توضیح داده خواهد شد.



۲-۲-۷- منوی Tools

این منو سایر پنجره های وبتز را باز می کند.

✓ : Scene Tree

پنجره درختی وبتز را باز می کند که در این پنجره می توان جهان مجازی را ویرایش کرد . همچنین با دوبار کلیک روی هر شی در پنجره اصلی وبتز پنجره Scene Tree همراه با انتخاب نام آن شی باز می شود.

✓ **Text Editor** :

پنجره گزارش وبتز را باز می کند. این پنجره برای ویرایش و تفسیر کدهای کنترلی بکار می رود.

✓ **Console** :

پنجره گزارش وبتز را باز می کند. این پنجره قابل ویرایش نیست و فقط خطاهای وبتز و خروجی های چاپی برنامه کنترلی را نشان می دهد.

✓ **Upload to e-puck robot** :

این گزینه به شما اجازه می دهد تا یک ارتباط بلوتوثی با ربات e-puck بر قرار کنید و آن را باز گذاری کنید.

✓ **Preferences :**

در این گزینه تنظیمات و اولویت های وبتز قرار دارد که شامل دو تب است:

۱- General :

✓ **Language** :

آن زبان منوهای وبتز را انتخاب می کند و البته نیازمند دوبار اجرا کردن برنامه است.

✓ **Startup mode :**

این گزینه اجازه انتخاب حالت شبیه سازی را هنگام باز کردن وبتز را می دهد (توقف ، اجرا ، سریع)

✓ **Editor font :**

فونت مورد استفاده در ویرایشگر متن وبتز را می دهد.

✓ **java command :**

دستورات جاوا برای شروع موتور مجازی جاوا را معین می کند که برای لینوکس و مکینتاش باید java و برای ویندوز باید javaw.exe باشد.

۲- Rendering :

بعضی جنبهای نمایشی پنجره شبیه سازی را بر عهده دارد. که با زدن تیک آنها فعال می شوند:

✓ **Display global coordinate system :**

سیستم مختصات جهانی را در صفحه شبیه سازی نمایش می دهد (پایین سمت راست پنجره) . در سه رنگ قرمز ، سبز و آبی که به ترتیب معرف محورهای x , y , z است.

✓ **Display sensor rays :**

برای نمایش پرتوی سنسورها در رباتهای انتخاب نشده است (پرتوهای رباتهای انتخاب شده همواره نشان داده می شوند). پرتوی سنسور فاصله یاب با یک خط قرمز کشیده می شود (که از نقطه بر خورد با مانع به بعدش سبز رنگ می شود) و پرتوهای سنسور نور با خط زرد رنگ نشان داده می شوند.

✓ **Display device axes :**

این گزینه نمایش محورهای سرو موتور و متصل شونده ها را فعال می کند . محور سرو موتور با یک خط چین سیاه رنگ ضخیم مشخص می شود و محور متصل شونده ها با خط های سبز و آبی بیانگر محور z,y کشیده می شود و خطوط سیاه نشان دهنده خط ترازی است که هنگام اتصال روی هم قرار می گیرند (عمل بار اندازی).

✓ **Display camera frustums :**

با فعال کردن این گزینه محدوده دید هر دوربین بصورت یک هرم سر بریده با خطهایی به رنگ صورتی نشان داده می شود.

✓ **Display lights :**

گره Point Light یا چراغ را با چندین خط زرد رنگ همرس نشان میدهد که بیانگر چشمه نور است.

✓ **Coordinate system size** :

اندازه محورهای مختصات محلی (بدنه) واقع در مرکز هر جسم صلب Solid nodes را معین می کند (موقعی که انتخاب شده باشد) . برای هر گره Solid دو مرکز می تواند نشان داده شود اول مرکز هندسی (که بوسیله فیلدهای جابه جایی و دوران هر گره صلب معین می شود) و دوم مرکز جرم که بوسیله

گره Physics معین می شود و در حالت پیش فرض این دو مرکز روی هر قرار دارند تا جسم از نظر همان اینرسی متعادل باشد.

✓ **Axis size :**

کنترل نمایش طول محور سرو موتورها را بر عهده دارد.

۲-۲-۸- منوی Wizard

برای شروع یک پروژه جدید یا یک برنامه کنترل جدید از این منو استفاده می شود ، شامل :

✓ **New Project Directory :**

این گزینه نخست به شما اعلام می کند که یک پوشه جدید در زیر مجموعه پوشه پروژه وبتز بسازید یا انتخاب کنید سپس در این پوشه ، وبتز چهار پوشه مورد نیاز پروژه جدید را می سازد.

این پوشه ها برای ذخیره سازی فایل World و کنترل ربات و سایر موارد لازم می باشند و شامل پوشه های worlds , Controllers , Protos , Plugins می باشد. این پوشه ها ابتدا خالی هستند و وبتز این پوشه ها را هنگام ذخیره سازی فایلهای به خاطر می آورد.

✓ **New Robot Controller :**

این گزینه را بعد از ساختن پوشه های پروژه به منظور ساختن پوشه کنترل و فایل make file درون آن باید انتخاب کرد .

بعد از انتخاب این گزینه وبتز پنجره انتخاب زبان برنامه نویسی را نشان می دهد که یک زبان از میان گزینه های C , C++ , Java , matlab , urbi باید انتخاب کرد سپس نام کنترلر جدید را سوال می کند و آن را در زیر مجموعه پوشه کنترل قرار می دهد و در نهایت فایل های لازم را می سازد.

✓ **New physics Plugin… :**

این گزینه در صورت نیاز به محیط فیزیکی جدید یک پوشه جدید فیزیک در زیر مجموعه پوشه Plugin می سازد و فایل های مورد نیاز را در آن قرار می دهد. توجه کنید که محیط فیزیکی جدید نیز به

صورت برنامه نویسی و استفاده از دستورات خاص موتور دینامیکی ode می باشد. این محیط فیزیکی می تواند موارد اضافی مانند آب و باد و سایر عوامل را به محیط ربات بیفزاید.

۲-۲-۹- منوی Help :

تنها گزینه قابل بیان در این منو گزینه Webots Guided Tout است که رباتهای ساخته شده در قسمت sample وبتز را به صورت پشت سر هم نشان میدهد. سایر گزینه ها راهنمای برنامه ، نحوه دسترسی به سایت سازنده و ثبت وبتز هستند.

۲-۳- انتخاب کردن اشیا

برای انتخاب هر شی با یک بار کلیک چپ آن شی مشخص می شود و اگر دارای مرز باشد مرز آن هم نمایان می شود و همزمان گره مرتبط با آن در پنجره درختی پر رنگ می شود . با دوبار کلیک کردن روی ربات پنجره رباتی نیز نمایان می شود و همچنین اگر پنجره درختی نیز بسته شده باشد با این کار ظاهر می شود.

۲-۴- جابه جایی ، دوران ، زوم و تغییر دید به کمک موس :

برای **جا به جایی** افقی هر شی در وبتز کافیست آن را آن را با کلیک چپ انتخاب کرد به صورتی که مرز سفید رنگ آن نمایان شود سپس دکمه shift صفحه کلید را پایین نگه داشته و موس را حرکت دهید در این حال شی با موس حرکت می کند.

برای **دوران** مانند جابه جایی عمل کرده بجز اینکه کلیک راست را و دکمه shift را همزمان پایین باید نگه دارید و موس را حرکت دهید .

در حالت اول دوران فقط حول محور y است ولی اگر کلیک راست موس را رها کرده و دکمه شیفت را یکبار دیگر سریع فشار دهید محور x ، محور دوران خواهد شد و اگر یکبار دیگر سریع فشار دهید محور z خواهد شد در همه این حالت بعد از تغییر محور دوران با کلیک راست می توان شی را حول محور دوران چرخاند.

برای **جا به جا کردن کل جهان** مجازی و صفحه نمایش کلیک کرده و دکمه ctrl را نگه داشته و موس را حرکت دهید. این کار با نگه داشتن کلیک سمت راست موس به تنهایی و حرکت آن نیز می توان انجام داد.

برای **انتقال عمودی** یا تغییر ارتفاع یک شی کافیست پس از انتخاب آن دکمه shift را نگه داشته و دکمه وسط موس را بچرخانید یا جفت کلیک کنید و موس را عقب یا جلو ببرید.

برای **چرخاندن زاویه** دید حول یک شی آن را انتخاب کرده و موس را حرکت دهید.

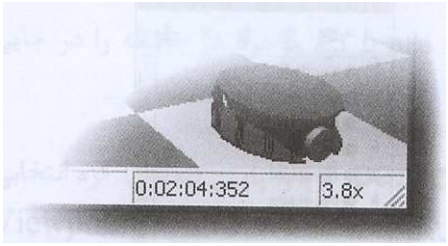
برای **زوم کردن** روی یک شی بوسیله چرخاندن دکمه موس این کار انجام می شود یا جفت کلیک کردن و عقب و جلو کردن موس . اگر در این حالت موس را افقی حرکت دهید دوربین دید شما حول نقطه دید می چرخد.

۲-۵- سرعت سنج و زمان مجازی

سرعت سنج (شکل ۲-۵) که در پایین و سمت راست پنجره شبیه سازی قرار دارد سرعت شبیه سازی کامپیوتر شما را نسبت به جهان واقعی نمایش می دهد. به عبارت دیگر این عدد سرعت زمان مجازی را نمایش میدهد.

اگر مقدار سرعت سنج عدد ۲ باشد ، معنی آن این است که کامپیوتر شما ربات و محیط شبیه سازی را دو برابر سرعت واقعی آن اجرا دارد می کند. این اطلاعات برای حالت سریع شبیه سازی نیز مفید است.

شکل ۲-۵

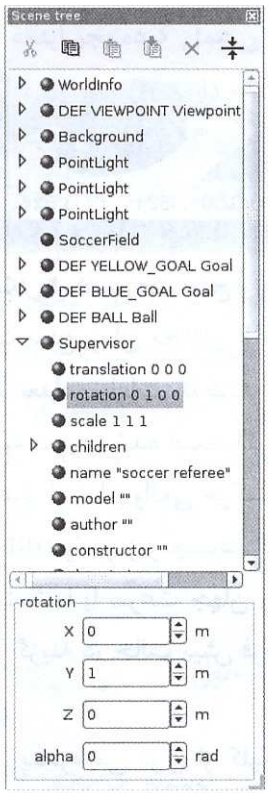


در سمت چپ سرعت سنج زمان مجازی با فرمت H: MM:SS:MMM نمایش داده می شود. H تعداد ساعات گذشته ,MM دقیقه ، SS ثانیه و MMM تعداد میلی ثانیه سپری شده است. اگر سرعت سنج بیشتر از یک را نشان دهد زمان مجازی تند تر از مان واقعی حرکت خواهد کرد.

اگر فیلد runreal time را در زیر مجموعه گره اول یعنی worldinfo به حالت true در بیاورید ، ربات شما با سرعت جهان واقعی و با گذشته زمان واقعی حرکت خواهد کرد که این گزینه در حالت پیش فرض برنامه غیر فعال است.

۲-۶- پنجره درختی

برای ظاهر شدن این پنجره می توان از کلید دوباره یا از منوی Tools استفاده کنید . این پنجره شامل اطلاعاتی است که جهان شبیه سازی شده را توصیف می کند ، مانند ربات و محیط و خصوصیات گرافیکی آنها ، ساختار پنجره درختی وبتز مانند ساختار VRML97 است که از لیست گره ها تشکیل یافته و هر گره شامل فیلدهایی می باشد و فیلدها در بر گیرنده مقادیر متغیری از انواع متنی و عددی یا سایر گره ها هستند.



شکل ۲-۶ . پنجره درختی

۲-۶-۱- دکمه های پنجره درختی:

Cut : یک گره انتخاب شده را جای خود جدا می کند.

Copy : گره یا فیلد انتخابی را کپی می کند.

Paste : فیلد یا گره موجود در حافظه را در جایی که انتخاب کرده اید می چسباند.

Paste after : گره را پایین (بعد از) گره انتخابی می چسباند. توجه داشته باشید که سه گره اول (Backrtound , World Info , View pint) قابل کپی و جدا کردن نیستند. فقط یک گره از هر کدام از این گره ها باید در هر جهان وبتز باشد.

delete : گره یا فیلد انتخابی را حذف می کند.

Reset to default : این دکمه مقدار فیلد را به مقدار اولیه خود بر می گرداند.

Transform :این دکمه نوع بعضی از گره ها را به گره دیگر تغییر می دهد. مثلا گره ربات را به گره ناظر تغییر می دهد.

insert after : دقیقاً یک گره بعد از گره فعلی قرار می دهد. با زدن این دکمه یک پنجره انتخاب باز می شود که شامل می توانید از لیست گره ها یکی را انتخاب کنید. در هر انتخاب فقط می توان گره های متناسب با آن را انتخاب کرد.

New node : این دکمه اجازه اضافه کردن یک گره جدید در جای خاص را می دهد (یا گره children یا در لیست گره های اصلی).

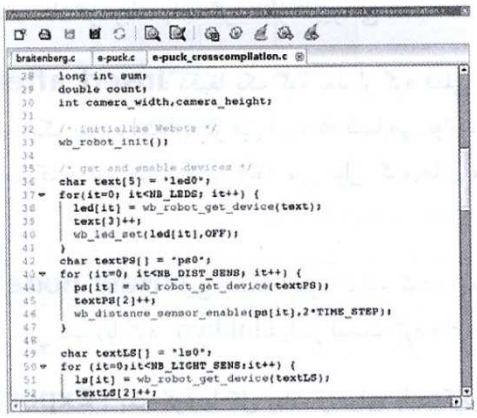
Export : یک گره یا کل ربات را برای وارد کردن به برنامه های دیگر به صورت فایل wbt . یا wrl ذخیره می کند.

Import : گره هایی را که از قسمت قبل صادر شده یا از برنامه های مدل سازی دیگر با پسوند wrl. ذخیره شده باشند را وارد جهان مجازی وبتز می کند و در آخر لیست گره ها قرار می دهد.

Help : این دکمه یک صفحه وب راهنما مربوط به گره انتخابی شما را باز می کند.

۲-۷- ویرایشگر کدهای برنامه نویسی

این ویرایشگر برای نوشتن کدهای برنامه نویسی و تفسیر و خطایابی آنهاست و کدهای هر زبان را به زنگ خاصی نشان می دهد و دستورات وبتز را نیز به رنگ سبز و به صورتی نمایش می دهد که به محض نوشتن اول دستور بقیه آن پیشنهاد می شود و نیازی به تایپ کامل آنها نیست . شکل ۲-۷



شکل ۲-۷ ویرایشگر کدها

دکمه های این پنجره در قسمت Edit , Build توضیح داده شد. خروجی چاپی این ویرایشگر در پنجره گزارش نمایان می شود وخطاهای برنامه را به رنگ قرمز با ذکر سطر خطا مشخص می کند با دوبار کلیک بر خطا ، خط برنامه مربوط به آن پر رنگ می شود.

ترجمه زبان برنامه نویسی و خطایابی کدها و دستورات نوشته شد در ویرایشگر را با کمک فایل Make file که باید در پوشه کنترل ربات با گزینه new controller در منوی ویزارد کپی شده باشد انجام می دهد.

توجه کنید این فایل را می توان از سایر پوشه های کنترلی واقع در رباتهای مثال وبتز نیز کپی کنید. این ویرایشگر فقط زبانهای java , C++ , , python ترجمه یا کامپایل می کند.

پروژه هایی با چندین فایل برنامه نویسی

وبتز پروژه هایی با چندین فایل برنامه نویسی را نیز پشتیبانی می کند. در زبانهای c , c++ نام فایل منبع باید در لیست فایل make file پروژه آمده باشد. برای مثال اگر پروژه شما دارای چندین فایل cpp یا cc . است یکی از این خطها را به make file پروژه تان اضافه کنید.

CPP _ SOURCES = my _ first _ file. Cpp

My_ second _ file. Cpp my_last _file.cpp

…

CC_SOURCES = my_first_file.cc my_second_file.cc

My_last_file.cc

در جاوا نیازی به این کار نیست چون javac به طور خودکار فایل های متعلق به پروژه را پیدا می کند.

۲-۸ ویرایشگر حرکت Motion Editor

این پنجره برای طراحی حرکات رباتهای مفصل دار است که می خواهند حرکتهایی را به ترتیب پشت سر هم اجرا کنند . برای مثال رباتهای انسان نما.

حرکت حاصل قابل ذخیره کردن است. فایل های حرکت در حین اجرای شبیه سازی دوباره اجرا می شوند. ویرایشگر حرکت برای کنترل چرخهای ربات Differential Wheels قابل استفاده نیست.

ویرایشکر حرکت مستقیما به موتور شبیه سازی متصل می شود و این کار چندین مزیت دارد :

نخست ، طراحی حرکات فیزیکی عملی را نسبتا آسان می کند.

دوم ، طراحی حرکاتی که با محیط در تعامل هستند را امکان پذیر می کند.

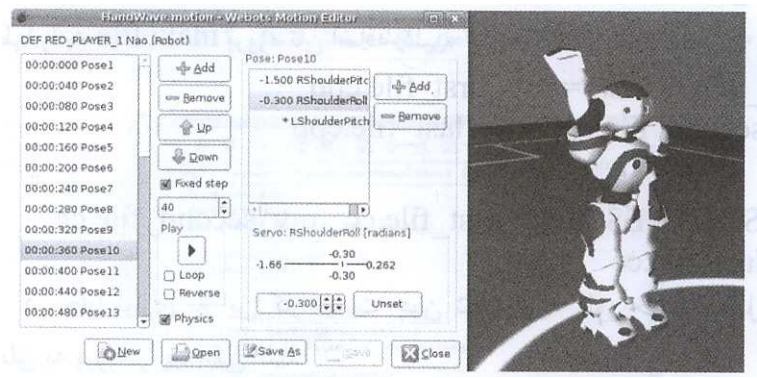
برای مثال اگر می خواهید حرکات یک دست گیرنده را طراحی کنید این کار به آسانی با اضافه کردن شییکه باید گرفته شود به محیط امکان پذیر خواهد بود.

۲-۸-۱- استفاده از پنجره ویرایشگر حرکت

به منظور استفاده از پنجره motion editor کافیست آن ربات را با موس انتخاب کرده و از منوی Robot گزینه Motion Editor را انتخاب کنید. این پنجره از دو قسمت تشکیل شده است.

سمت چپ برای کار با وضعیت های متوالی سمت راست برای انتخاب و اصلاح موقعیت مفاصل در هر وضعیت ، هر وضعیت مانند گرفتن یک عکس از ربات در یک حالت خاص است. شکل ۲-۸

شکل ۲-۸



لیست سمت چپ زمانهای مطلوب و نام هر وضعیت را نشان می دهد که این وضعیتهای با زدن دکمه Add اضافه می شوند و زمان به فرمت دقیقه ، ثانیه و میلی ثانیه نوشته می شود.

نام وضعیت ها در هنگام شبیه سازی در نظر گرفته نمی شوند ولی برای ذخیره سازی مفید هستند. دکمه های Add , Down , Up , Remove به ترتیب برای اضافه کردن ، حذف کردن و بالا و پایین کردن وضعیتها استفاده می شود.

Fixed step زمان بین وضعیت های را در صورت لزوم ثابت می کند. اگر این تیک زده شود ویرایشگر حرکت گام زمانی را به صورت خودکار مدیریت می کند . در غیر این صورت باید به صورت دستی وارد شود. فیلد عددی کنار آن اندازه گام زمانی را به صورت ثابت در واحد میلی ثانیه قرار می دهد.

اگر این عدد تغییر کند به صورت خودکار در هر وضعیت اثر می گذارد و این راهی آسان برای اصلاح سرعت حرکت نمایش ربات است.

دکمه play نمایش حرکت را آغاز می کند. این کار یک درخواستی را به موتورهای سرو ربات می دهد و این در خواست مانند اجرای این دستور اجرا (run) باشد .

در غیر اینصورت ربات حرکت نخواهد کرد. گزینه Loop در صورتی که حرکت بخواهد تکرار شود باید تیک زده شود. گزینه Reverse در صورتی که حرکت بخواهد از آخر به اول تکرار شود باید تیک زده شود.

گزینه Physics شبیه سازی فیزیکی را به جای شبیه سازی سینماتیکی ربات قرار می دهد.

سمت راست ویرایشگر حرکت بر روی وضعیتی که در سمت چپ انتخاب شده است عمل می کند و اجاز می دهد که یک یا چندین مفصل را انتخاب و اضافه به این وضعیت کنید. توجه کنید که علامت * بیان می دارد که این مفصل برای این وضعیت تعریف نشده است.

۲-۹- آشنایی با گره ها در وبتز

گره های وبتز که در اینجا معرفی می شوند از استاندارد VRML استفاده می کنند . اصولا گره های وبتز زیر مجموعه گره های VRML97 هستند. ولی گره ها و فیلدهای دیگر نیز مختص رباتیک در وبتز وجود دارد. برای مثال گره World Into و گره کره اضافه بر گره های VRML97 هستند.

وبتز برای شبیه سازی قسمت فیزیکی آن به موتور باز دینامیکی ODE بر می گردد. بنابراین توصیه می شود راهنمای ODE را برای فهمیدن این پارامترها مطالعه کنید که در سایت آن و سی دی همراه کتاب موجود است. همچنین برای ساختن محیط فیزیکی با ویژگی های متفاوت به دستورات ODE نیاز دارید.

۲-۹-۱- طرح کلی گره های وبتز

به هر شی موجود در نرم افزارهای مدلسازی گره می گویند که می تواند خاصیتهایی برای آن تعریف کرد. مثلا سنسورها ، موتورهای محرک ، اشیا مرزی ، شکلها ، چراغ ، ربات ، موانع، زمین و غیره ... بعضی گره ها زیر مجموعه بعضی دیگر قرار می گیرند و گره بالاتر از گره پایین ترش درست می شود و خواص آن را به ارث می برد. مثلا یک گره جسم صلب مثل چرخ می تواند زیر مجموعه یک سرو موتور شود.

هر خاصیت یک گره را که قابل تغییر است را فیلد گویند.

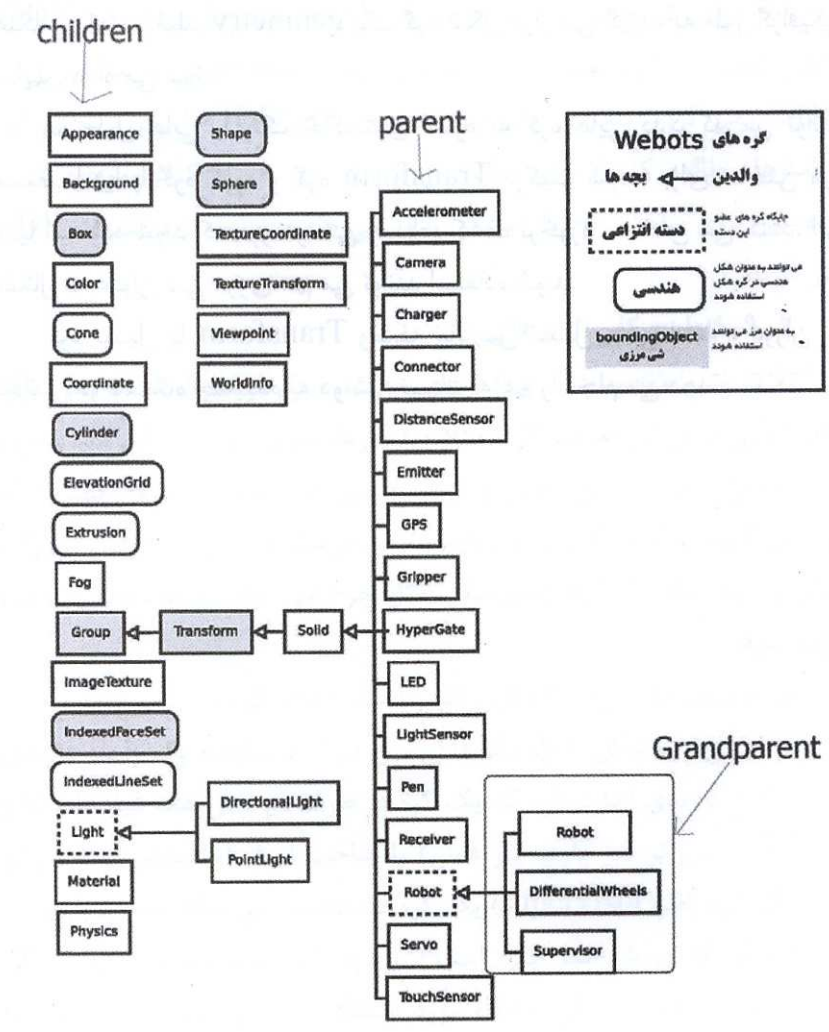
در این چارت شکل ۲-۹ ، یک فلش بین دو گره نماینده یک رابطه فرزندی است . رابطه فرزندی اشاره دارد که یک گره در طرف دم فلش هم فیلدها و توابع یک گره اساسی تر (در طرف سرفلش) را صاحب است (والدینش است) . برای مثال یک گره ناظر (Supervisor) یک گره فرستنده را می تواند صاحب باشد.

گره ای که دورش خط چین است اشاره به یک دسته انتزاعی دارد که گره های عضو این دسته در آن جایگاه مورد استفاده قرار می گیرند. مانند نور که شامل چراغ و نور مستقیم خورشید است .

مثال دیگر دسته ربات است که شامل یک ربات دلخواه ، یک ناظر و یک دو چرخ دیفرانسیل دار است که همگی ربات محسوب می شوند. مستطیل هایی با گوشه گرد بیانگر گره های هندسی هستند ، هنگامی که در فیلد geometry یک گره شکل قرار می گیرند به طور گرافیکی نمایش داده می شوند.

مستطیل هایی با رنگ خاکستری اشاره به گره هایی دارد که می توانند مستقیما (یا با گره گروه و گره Transform ترکیب شوند) برای ساختن مرز اشیا استفاده شوند که مرز دو شی را برای کشف بر خورد مشخص می کنند. این اشکال به عنوان شی مرزی هم می توانند استفاده شوند.

گره تبدیل یا Transform وظیفه ماتریس تبدیل یعنی انتقال ، دوران و تجانس دو دستگاه چسبیده به دوشی نسبت به هم را انجام می دهد.



فصل سوم :

برنامه نویسی و کنترل

۳-۱- طرح کلی برنامه

هر برنامه کنترل ربات از ترتیب یکسانی ساخته شد است. دستور اولیه wb_robot_init() برای شروع کار ربات لازم است . سپس باید وسایل همراه ربات را معرفی نمود و آنها را روشن نمود.

کار بعد این است که یک حلقه نامحدود که در هر گام زمانی تکرار می شود را ایجاد کرد که این گام زمانی با دستور wb_tobot_step() معرفی می شود که بر حسب میلی ثانیه است.

به طور کلی کار حلقه ی نامحدود این است که اطلاعات را از سنسورها می گیرد ، آنها را پردازش می کند و خروجی لازم را به موتورها می دهد و نهایتاً دستور wb_tobot_step() را فرا می خواند .

دستور wb_tobot_cleanup() در پایان برنامه باید درج شود تا خروج از برنامه با پاک کردن حافظه ها انجام گیرد. در پوشه Projects / Packages مثالهای اجرایی برای آشنایی با برنامه نویسی با انواع زبانها قرار دارد.

۳-۲- اجرای برنامه از دیدگاه Webots

در هر ربات ، وبتز به پوشه کنترلر پروژه نگاه می کند تا فایل کنترلی مورد نظر را بیابد. برای مثال اگر برنامه کنترلی نام آن simple باشد ، وبتز سعی می کند تا نخست فایل simple.exe را اجرا کند.

اگر چنین فایلی پیدا نشد دنبال فایل sample.class می گردد و آن را به عنوان یک برنامه جاوا آغاز می کند. اگر این فایل نیز پیدا نشد دنبال فایل simple. Jar می گردد و در نهایت فایل simpe.py را جستجو می کند.

اگر این فایل نیز نباشد آنگاه Webots دیگر قادر نیست برنامه کنتری را آغاز کند و از برنامه کنترلی void به عنوان پیش فرض استفاده می کند. همه دستورات خاص وبتز ، روالی مانند شکل ۴-۱ دارند.

۳-۳ – کنترلهای همزمان و غیر همزمان

هر نوع رباتی ، کنترل آن یا همزمان است یا غیر همزمان . وبتز قبل از اینکه زمان شبیه سازی را جلو ببرد منتظر برنامه کنترلی همزمان می ماند. بنابراین یک برنامه کنترلی غیر همزمان ممکن است دیرتر اجرا شود.

توصیه می شود از کنترلهای همزمان استفاده کنید و تنها زمانی از کنترل غیر همزمان استفاده کنید که محاسبات ربات بالا باشد و یا محدودیت کامپیوتری وجود داشته باشد و یا اینکه از یک شبکه شبیه سازی با چندین ربات استفاده می کنید که زمان تاخیرش نامعلوم است مانند شبکه اینترنت.

۳-۴- خواندن اطلاعات سنسورها

برای جمع آوری اطلاعات سنسورها ، سنسور باید چنین باشد .

۱- معرفی بشود :

که بوسیله wb_robot-get_device() انجام می شود که وسیله را بر اساس فیلد نامش آن (name) جستجو می کند. این کار فقط یکبار کافیست اجرا شود آن هم قبل از حلقه اصلی ، خود ربات نیز یک وسیله است ولی نیازی به معرفی شدن ندارد چون به طور پیش فرض معرفی شده است.

۲- فعال باشد (enabled) : که با دستوری مانند

Wb_distance_sensor_enable() فعال می شود . این کار نیز یکبار قبل از حلقه اصلی باید انجام شود. البته اگر قصد غیر فعال کردن آنها را دارید می توانید در حلقه اصلی این سنسورها را چندین بار فعال و غیر فعال کنید.

۳- اجرا شود (run) : که بطور خودکار توسط wb_robot _step() اجرا می شود.

۴- خوانده شود: در نهایت شما می توانید مقادیر سنسورها را با استفاده از دستور خاص خودش بخوانید مانند wb_distance_sensor_get_value() که در حلقه اصلی این دستورات قرار می گیرند.

۳-۵- کنترل محرک ها (موتورها)

یک محرک آسانتر از سنسور کنترل می شود. آنها نیاز به فعال شدن ندارند . برای کنترل یک محرک باید :

۱- **معرفی بشود:**

این کار با دستور wb_robot_get_device() اجرا می شود که محرک را با استفاده از فیلد نامش جستجو می کند و فرامین را به آن باز می گرداند . این کار فقط باید یکبار قبل از حلقه اصلی انجام شود.

۲- **تنظیم شود:**

این کار با دستور تنظیم سرعت یا موقعیت یا شتاب و غیره انجام می شود مانند : wb_differential_wheels_set_speed() این دستورات در حلقه اصلی درج می شوند.

۳- **اجرا شوند:**

این کار در طول حقله و بعد از دستور wb_robot_step() به طور خودکار اجرا می شود.

۳-۶- برنامه کنترلی ناظر (supervisor)

ناظر به عنوان یک ابر ربات است ناظر قادر است هر چیزی را که یک ربات انجام می دهد انجام دهد و حتی بیشتر ، این ویژگی ها وقتی مفید است که بخواهیم پیامی را به ربات بفرستیم یا با کمک گره های فرستنده (Reciver) و گیرنده (Emitter) پیامی را از او بگیریم .

علاوه بر این یک ناظر کارهای جالب دیگری نیز می تواند انجام دهد مثال اداره کردن پنجره درختی در طول اجرای شبیه سازی و جابه جا کردن ربات یا هر شی بی حرکت دیگری یا نقطه دید را عوض کند ، رنگ اشیا را تغییر دهد ، چراغی را خاموش یا روشن کند.

همچنین می تواند مختصات یک ربات را دنبال کند که برای ضبط مسیر یک ربات مفید است. ناظر نقش یک انسان ناظر بر محیط و ربات را بازی می کند که لزوما نیازی نیست در هر پروژه های باشد. ناظر می تواند از صحنه نمایش عکس نیز بگیرد و آن را با فرمت png یا jpg ذخیره کند یا حتی فیلم بگیرد.

۳-۷- انتقال برنامه به ربات واقعی

در شبیه سازی رباتهای متحرک ، اغلب مفید است که نتایج را به یک ربات واقعی انتقال دهیم. Webots چنان طراحی شده است که این توانایی را دارد.

محیط شبیه سازی می تواند با ربات واقعی ارتباط بر قرار کند. وبتز همچنین دارای سیستم انتقال به چندین ربات واقعی به صورت آماده است ، نظیر :

رباتهای آموزشی LEGO Mindstorms Aibo , Hemisson , Khepera , e-puck و غیره .
در این قسمت یاد می گیرد که چطور سیستم انتقال خودتان به ربات واقعی را انجام دهید.

از آنجایی که محیط شبیه سازی یک تقریب از فیزیک واقعی ربات است ، بعضی میزان سازی ها برای انتقال به ربات واقعی همیشه مورد نیاز است. این میزان سازی تلاش خواهد کرد تا مدل شبیه سازی با ربات واقعی هماهنگی بهتری داشته باشد.

۳-۷-۱- کنترل از راه دور

اغلب آسانترین راه انتقال استفاده از سیستم راه دور است که این سیستم اطلاعات را از ربات فرا می خواند و دستورات را می فرستد. یک سیستم کنترل از راه دور به سادگی با نوشتن دستورات وبتز در یک کتابخانه کوچک انجام می گیرد.

باری مثال ، شما ممکن است دستور wb_differential_wheels_set_speed() را برای فرستادن سرعت خاصی به چرخهای ربات بخواهید انجام دهید ، این دستور می تواند توسط پورت سریال کامپیوتر فرستاده شود.

شما احتمالا به یک واحد تبدیل نیز احتیاج دارید زیرا ربات شما ممکن است از واحد واقع در وبتز برای اندازه گیری استفاده نکند.

ایجاد یک کتابخانه دلخواه: هنگامی که یک سری دستورات به زبان c همراه با دستورات وبتز می نویسید ، باید آنها را با خروجی و ورودی های ربات واقعی هماهنگ کنید.

آنگاه دیگر می توانید از برنامه کنترلی وبتز بدون تغییر حتی یک خط نیز دوباره استفاده کنید و به جای پیوند با کتابخانه دینامیکی کنترلر وبتز می تواند با دستورات زبان c خود پیوند بر قرار کنید.

۳-۷-۲- انتقال کد به پردازشگر ربات

با استفاده از سیستم cross-compiliation شما می توانید برنامه کنترلی وبتز را دوباره برای میکروپروسور تعبیه شده در ربات واقعیتان تفسیر کنید.

بنابراین کدهای نوشته شده در وبتز دوباره در ربات واقعی استفاده می شوند و دیگر نیازی به یک کامپیوتر دائمی متصل به ربات حتی توسط کنترل از راه دور هم نیست . این فقط زمانی امکان پذیر است که پردازشکر هایی که فقط زبان های c , c++ , java یا python را بپذیرد و برای پردازشگرهایی که فقط زبان اسمبلی یا زبانهای خاص دیگر را قبول می کنند امکان پذیر نیست.

وبتز به طور پیش فرض دارای کدهای سیستم cross-compilation برای رباتهای e-puck , Hermisson می باشد. نمونه های آن در پوشه robots وجود دارد.

۳-۸- برنامه نویسی کنترلی در وبتز به URBI

URBI (Universal Robotic Body Interface) مخفف رابط جهانی اجسام رباتیک است که توسط COSTA توسعه یافت و اطلاعات بیشتر درباره آن را می توان در یک سایت یافت: http://www.urbiforge.com

URBI هنوز یک زبان ساده و قدرتمند است که به راحتی با رباتهای واقعی ارتباط بر قرار می کند. این زبان یک ساختار مشتری و سرویس دهنده دارد. ربات سرویس دهنده یا سرور هست و برنامه کنترلی مشتری می باشد. با این حال می توان در یک ماشین مشتری را به عنوان سرور اجرا کرد یا سرور را بدون مشتری اجرا کرد.

برای استفاده از URBI باید آن را در کامپیوتر نصب کنید. این سایت دانلود آن می باشد

ww.gostai.com. در پوشه ای با این مسیر projects / packages/urbi/worlds مثالهای بسیاری از URBI وجود دارد.

۳-۹- ارتباط Webots با نرم افزار Matlab

برای برنامه نویسی به این زبان باید نرم افزار Matlab روی کامپیوتر شما نصب شده باشد (زبان python نیز بدین صورت است).

اگر فردی نخواهد برنامه کنترلی ربات خود را با زبانهای برنامه نویسی رایج مثل c یا c++ یا java بنویسد ، این امکان وجود دارد که وبتز با نرم افزارهای سه قسمتی مانند Matlab یا LabView ارتباط بر قرار کند. این ارتباط از طریق پروتکل TCP/IP انجام می گیرد.

راه دیگر استفاده از رابطه برنامه نویسی به زبان C نر افزار Matlab است. یک مثال از استفاده از پروتکی TCP/IP در پوشه worlds / projects / robots / khepera وجود دارد. با نام Khepera tepip .wbt

مزیت استفاده از این پروتکل آن است که می توان چندین ربات را در یک جهان مجازی داشته باشید و با استفاده از پروتکل TCP/IP می توان چندین رابطه کنترلی داشته باشید که هر یک از یک پورت جداگانه استفاده کنند.

برای این کار باید نامهای مختلفی به هر ربات داده و با دستور robot_get_name() از هر کدام استفاده کنید. البته این روش محدودیتهایی نیز دارد.

ویرایش استاندارد Matlab ممکن است جعبه افزار TCP/IP را نداشته باشد. این جعبه افزار که TCP/UDP/IP Toolbox 2.0.5 نام دارد در سی دی وبتز وجود دارد ، همچنین می توان آن را از سایت نرم افزار Matlabنیز دانلود کرد. برای کسانی که با جاوا کار می کنند این رابطه به زبان جاوا نیز وجود دارد.

راه دیگر استفاده از رابط زبان c نرم افزار Matlab است که می تواند با برنامه کنترلی وبتز ارتباط بر قرار کند و هر گونه داده ای میان آنها رد و بدل شود برای مثال تصاویر از دوربین وبتز اطلاعات بیشتر را می توان در راهنمای Matlab در بخش ، فراخوانی Matlab از طریق c یا فرترن ، ببینید.

۳-۱۰- ابزارهای جانبی وبتز

وبتز می تواند بوسیله کاربر سه نوع برنامه به آن اضافه شود:

Physics, fast 2D, Sound

۳-۱۰-۱- ابزار فیزیک

این برنامه جانبی برای اضافه کردن دستورات موتور باز دینامیکی ODE به محیط فیزیکی اولیه وبتز است. مثال با اضافه کردن نیروی خاص می توان محیطی جدید را شبیه سازی کرد مثلا نیروی ارشمیدس برای آب ، یا نیروی درگ باد بر تمام اشیا وارد شود.

۳-۱۰-۲- ابزار محیط دو بعدی سریع Fast2D

با استفاده از این الحاق می توان محیط سه بعدی را دور زد و محیط الگوریتم دو بعدی ساده برای کنترل حرکات دو بعدی رباتها نوشت . وبتز یک ویرایش اجرایی از یک ابزار جانبی Fast 2D به نام enki ارائه می دهد که اطلاعات درباره آن را می توان از سایت http://home.gna.org/enki/ بدست آورید.

۳-۱۰-۳- ابزار جانبی صدا

این الحاق یک رابط برنامه نویسی برای الگوریتم انتشار صدا ارائه می دهد. شبیه سازی صدا بر اساس نمونه صدایی است که از گره بلند گو به گره میکروفن پخش می شود. وبتز یک ویرایش از این برنامه جانبی صدا به نام swis2d را تهیه کرده است. اطلاع بیشتر درباره آن از اینجا می توان تهیه کرد:

http://swis.eprl.ch

وبتز برای این ابزارهای جانبی مثالهایی دارد.

فصل چهارم :

گره ها و دستورات آنها

۴-۱- گره ربات دو چرخ دیفرانسیلی Differential Wheels

این گره نوعی ربات است و با انتخاب آن دیگر نوع ربات خود را انتخاب نموده اید.

گره Differential Wheels برای مدل کردن رباتهایی با دو چرخ محرک مجزا که هر کدام جداگانه دارای یک موتور هستند. ، استفاده می شود. در این نوع ربات ساده فقط با دستور تغییر سرعت می توان چرخها را به حرکت واداشت.

این ربات به طور خودکار دنبال چرخهایش در فیلد children خود می گردد و آنها را کنترل می کند. این چرها باید گره Solid باشند و حتما باید فیلد نام یکی right wheel و دیگری left wheel باشد.

اگر این چرخها یافت شد وبتز به طور خودکار آنها را با سرعتی که در دستور wb_differential_wheels_set_speed() نوشته شده است ، می چرخاند.

حرکت واقعی ربات به دو روش متفاوت انجام می شود:

۱- **بر پایه مدل سینماتیکی:**

در این روش برای ربات فیزیکی وجود ندارد یعنی فیلد فیزیک آن خالی است (جرمی ندارد). دی این هنگام دیگر شکل ، اندازه و موقعیت چرخها مهم نیست ، زیرا که بدنه ربات با هم با یک سرعت خطی که حاصلضرب فیلد شعاع چرخ در سرعت دورانی تعریف شده در برنامه کنترلی ، به جلو حرکت می کند و دیگر دوران چرخها به چشم دیده نمی شوند. در اینجا طول محور چرخ نیز دارای اهمیت است.

در روش سینماتیکی ارتفاع اولیه ربات نسبت به زمین را باید به اندازه ای انتخاب کرد که بر خورد با زمین نداشته باشد تا ربات حرکت کند. زیرا وجود چرخ الزامی نیست.

۲- **بر پایه مدل فیزیکی :**

در این روش برای ربات فیزیکی تعریف شده است و فیلد فیزیک دارای گره فیزیک است . در این حالت ربات بر اساس موتور فیزیکی ODE حرکت می کند و حرکت ربات توسط نیروی جلو برنده اصطکاک چرخ با زمین بدست می آید در نتیجه نوع چرخها و نقاط تماس مهم است و حضور چرخ الزمامی است.

البته چرخ هر شکلی می تواند داشته باشد اما باید با زمین در تماس باشد. در این روش که به واقعیت نزدکتر است دوران چرخ با چشم دیده می شود . فیلدهای شعاع و طول محور چرخها در این حالت نادیده گرفته می شوند و آنچه مدل می شود به عنوان چرخ محسوب می شود.

مبدا دستگاه مختصات ربات در وسط محور دو چرخ و تصویر شده بر روی زمین است. محور چرخ ربات در راستای محور x قرار می گیرد. محور z + اشاره به عقب ربات دارد و محور z– جلوی ربات را نشان می دهد و نتیجتا محور y در راستای ارتفاع ربات است.

جدول فیلدها

دامنه فیلد	مقدار پیش فرض	نام فیلد	نوع متغییر
)∞[0,	0	MotorConumption	SFFloat
)∞[0,	0.1	axleLength	SFFloat
)∞[0,	0.01	wheelRadius	SFFloat
)∞[0,	10	maxSpeed	SFFloat
	10	maxAcceleration	SFFloat
	1	speedUnit	SFFloat
	0.1	slipNoisc	SFFloat
)∞[0,	-1	Encoder Noise	SFFloat
	-1	encoderResolution	SFFloat
)∞[0,	0.3	maxForce	SFFloat

تعریف فیلدها:

- **Motor Consumption :** مصرف انرژی موتور بر حسب وات را مشخص می کند.
- **Axle Length :** فاصله میان دو چرخ بر حسب متر است . این فیلد در مدل سینماتیکی باید تعریف شود ولی در مدل فیزیکی نادیده گرفته می شود.
- **wheel Radius :** شعاع چرخ به متر است. هر دو چرخ دارای یک شعاع یکسان هستند. این فیلد برای مدل هندسی باید مشخص شود ولی در مدل فیزیکی نادیده گرفته می شود.
- **max Speed :** ماکزیمم سرعت دورانی چرخها را بر حسب رادیان بر ثانیه مشخص می کند (rad/s)
- **Max Acceleration :** ماکزیمم شتاب زاویه ای چرخها را بر حسب rad/s^2 مشخص می کند که فقط در مدل سینماتیکی محسوب می شود.
- **speed Unit :** واحد عدد مورد استفاده در دستور wb_differential_wheels_set_speed () را مشخص می کند که عددی ضرب در ral/s است و به طور پیش فرض مقدار یک است.
- **slip Noise :** این عدد خطای لغزش چرخهاست که بر درصد بیان می شود. فرضا اگر ۱،۰ است یعنی مثبت و منفی ده درصد در هر گام زمانی حرکت لغزش وجود دارد.
- **encoder Noise :** در وبتز انکدر یا دوران شمار به صورت سنسوری مستقل وجود ندارد بلکه یک دستور از دستورات ربات دو چرخ دیفرانسیلی است:

(wb_differential_wheels_get_left_encoder) که در طول برنامه می تواند مقدار دوران را بدهد و مقدار اغتشاش آن در این فیلد مشخص می شود.

اگر۱- باشد انکدر غیر فعال می شود. اگر مقدار صفر را به این فیلد بدهیم ، آنکدر بدون هیچ اغتشاشی شبیه سازی می شود. در غیر اینصورت یک اغتشاش یکنواخت به مقدارهای انکدر اضافه می شود. در هر گام شبیه سازی ، یک مقدار در حال افزایش برای هر انکدر محاسبه می شود.

سپس قبل از اینکه به مقدار انکدر افزوده شود یک اغتشاش یکنواخت و تصادفی به این مقدار در حال افزایش اعمال می شود. این اغتشاش تصادفی مانند اغتشاش لغزشی (slip noise) محاسبه می شود.

هنگامی که ربات با یک مانع مواجه می شود اگر شبیه سازی فیزیکی نباشد ، چرخهای ربات نمی لغزند. ، نابراین مقادیر انکدر افزایش نمی یابند. این عمل برای اینکه برخورد ربات با یک مانع را تشخیص دهیم بسیار مفید است.

- **encoder Resolution :**

این فیلد مقدار افزایش انکدر را در هر رادیان مشخص می کند. مثلا اگر ۱۰۰ باشد یعنی اینکدر مقدارش ۶۲۸ بار در هر دور بیشتر می شود (یک دور کامل دایره ای ۲۸،۶ رادیان است) . مقدار ۱- (پیش فرض) یعنی انکرد همانند اغتشاش آن غیر فعال است.

- **max Force :**

معرف ماکزیمم گشتاور مورد استفاده برای چرخاندن هر چرخ است و در حال شبیه سازی فیزیکی استفاده می شود. این فیلد معادل پارامتر dParam FMax در مفصل لولایی موتور دینامیکی ODE است. این فیلد در شبیه سازی سینماتیکی نادیده گرفته می شود.

دستورات ربات دو چرخ دیفرانسیلی Differential Wheels

1. Wb_differential_wheels_set_speed

2. Wb differential wheels enable encoders

3. Wb differential wheels disable encoders

4. Wb_differential_wheels_get_left_encoders

5. Wb_differential_wheels_get_right_encoders

6. Wb_differential_wheels_set_encoders

کد دستور به زبان c

```
#include<webots/differential_wheels.h>

Void wb_differential_wheels_set_speed (double left, double right);

Void wb_differential_wheels_enable_encoders (int ms);

Void wb_differential_wheels_disable_encoders ();

Double wb_differential_wheels_get_left_encoders ();

Double wb_differential_wheels_get_right_encoders ();

Void wb_differential_wheels_set_encoders (double left, double right);
```

توصیف دستورات Differential Wheels

دستور ۱، به کاربر اجازه می دهد تا یک سرعت دورانی را برای هر کدام از چرخها مشخص کند. واحد این سرعت در فیلد Speed Unit ربات مشخص می شود. مقدار پیش فرض یک رادیان بر ثانیه است. بنابراین مقدار ۱۰ یعنی چرخهابا سرعت ۱۰ رادیان بر ثانیه می چرخند.

سرعت خطی ربات نیز از این سرعت ، شعاع چرخ و اختلال لغزشی محاسبه می شود. این نکته را بخاطر داشته باشید که در قسمت کد نویسی دستور مقادیر بر رنگ شده نوع متغیرها را مشخص می کنند و هنگام تایپ دستور در اینجا نوشته نمی شوند بلکه در قسمت تعریف متغیرها می آیند.

مثالا در اینجا void یعنی این دستور چیزی را به برنامه بر نمی گرداند بلکه فقط اعمال می کند و double یعنی left یک عدد اعشاری یا متغیری از این جنس باید باشد . توصیه می شود مقدمات زبان c را فرا بگیرید.

خط #include<webots/differential_wheels.h> به این معنی است که این وسیله فراخوانی شده و در ربات گرفته شده است. این دستور ها در اول برنامه نویسی در قسمت تعاریف وسایل و متغیرها می آید. دستورات در پایان کتاب به زبان Matlab و در راهنمای برنامه به زبانهای دیگر نیز آورده شده اند.

دستور ۲ و ۳ ، این دستور به کاربر اجازه می دهد تا یک دور شما یا انکدر افزاینده را برای هر دو چرخ فعال یا غیر فعال کند. این وسیله هر بار که چرخ بچرخد مقدارش افزایش می یابد. مقدار اضافه شده بستگی به دوران چرخ و دقت انکدر دارد که در فیلد encoder Resolution مشخص می شود.

انکدر در هر مدت زمانی یک اطلاعی از دوران چرخ می گیرد که این زمان در مقدار پرانتز بر حسب میلی ثانیه مشخص می شود.

توجه داشته باشید ، هنگامی که ربات با یک مانع مواجه می شود از آنجایی که چرخها نمی لغزند و دوران هم دیگر نمی توانند بکنند در نتیجه مقدار انکدر نیز افزایش نمی یابد و این راه خوبی برای تشخیص بر خورد ربات با مانع است.

دو دستور ۴ و ۵ مقدار انکدرهای چرخ چپ و راست را می خوانند. البته انکدری که قبلا فعال شده باشد. دستور شماره ۶ ، انکدر را با مقداری که برایش مشخص شده است دوباره راه اندازی می کند و انکدر دوباره از این مقدار شروع به کار می کند. ولی هیچگاه در دوران چرخ تاثیری نخواهد داشت.

۴-۲- گره جسم صلب solid (چه گره هایی می توانند مرز باشند)

گروهی از اشیاه هستند که می توان آنها را در جهان مجازی بوسیله موس نیز حرکت داد ، سنسورهای ربات و وسایل کشف بر خورد بوسیله مرز گره صلب عمل می کنند. جسم صلب در مکانیک جسمی را گویند که ذرات آن کنار هم تحت اثر نیرو و حرکت ثابت باقی بمانند. گره های زیر فیلدهای گره صلب را دارا هستند و خصوصیات آن را می توانند داشته باشند.

شتاب سنج ، شارژ کننده ، فرستنده دست گیرنده ، قلم ، ربات ، دوربین ، قطب نما ، سنسور فاصله یاب ، جی پی اس ، سرو ، سنسور تماسی ، چراغ LED

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض
SFSrting	Name	" "
SFSrting	Model	" "
SFSrting	Author	" "
SFSrting	Constructor	" "
SFSrting	Description	" "
SFNode	Bounding Object	NULL
SFNode	physics	NULL
SFNode	locked	FALSE

تعریف فیلدها

توجه کنید که فیلد Scale زیر مجموعه گره Transform , solid که معرف مقایس است همراه با [۱ ۱ ۱] باشد تا در کشف بر خورد مشکلی بوجود نیاید.

- **Name :** نام جسم صلب است که این نام در برنامه کنترلی توسط دستور wb_robot_get_divece () استفاده می شود.
- **model :** نام عمومی جسم است مانند صندلی (chair)
- **Autbor :** نام بوجود آورنده مدل شبیه سازی شده جسم صلب است.
- **Constructor :** نام شرکت یا فرد سازنده جسم واقعی است.
- **Description :** توضیح کوتاه در حد یک خط درباره جسم صلب است.
- **bounding Object :** این فیلد شی مرزی پیرامون جسم تعریف می کند که فقط برای تشخیص بر خورد از آن استفاده می شود. مرز جسم صلب فقط این اشکال می توانند باشند:

۱- گره مکعب (box)

۲- گره استوانه (cylinder) یک استوانه نه بسته ، استوانه ته یا سر باز نمی تواند مرز باشد.

۳- گره کره (sphere)

۴- گره رویه (indexedFaceSet)

۵- گره شکل ، هنگامی که شامل یکی از هندسه های فوق باشد.
۶- گره تبدیل مختصات (Transform) وقتی که فقط یک گره children داشته باشد که آن هم شامل یکی از موارد فوق باشد و مقیاس هم یک باشد.
۷- گره Group با چندین گره children که هر کدام باید شامل یکی از موارد بالا باشد.

در حالتی که فیلد فیزیک جسم صلب خالی باشد و مرز شامل فقط یک گره Transform باشد ، این گره تبدیل مختصات موقعیت مرکز جرم جسم را تعیین می کند. علاوه بر این اگر در گره فیزیک ، ماتریس اینرسی (فیلد inertia Matrix) تعریف شود ، آنگاه جهت گیری گره تبدیل مختصات در جهت گیری ماتریس اینرسی نیز تاثیر دارد.

در حالتی که شکل جسم ، رویه)IndexedFaceSet) باشد ، دو انتخاب برای مرز موجود است: اولین انتخاب یک رویه مربعی که یک صفحه را تشکیل می دهد که موتور کشف بر خورد ، آن را به عنوان یک صفحه بی نهایت بزرگ می شناسد.

این حالت در مثال ربات boebot.wbt در مسیر projects/robots/boebot/worlds استفاده شده است. انتخاب دو ، استفاده از یک رویه مثلثی است که یک مش بندی مثلثی (trimesh) را برای مرز بوجود می آورد.

این نوع رویه ها را به آسانی از نرم افزارهایی مدل سازی سه بعدی بعد از مش بندی مثلثی می می توان وارد وبتز کر یا در محیط وبتز ساخت. این روش در مثال ربات aibo در مسیر projects/robots/aibo/worlds استفاده شده است.

شی مرزی یا همان مرز به طور خودکار برای محاسبه ماتریس اینرسی جسم صلب نیز استفاده می شود. لطفا توجه داشته باشید که مرکز جرم جسم همواره در مبدا مختصات جسم ، باقی می ماند ، بدون توجه به نوع و شکل مرز ، که این مبدا بوسیله فیلد انتقال و دروان مشخص می شود.

اگر فیلد مرز خالی باشد هیچ کشف بر خوردی و اثر فیزیکی وجود نخواهد داشت. مرکز جرم فقط بوسیله فیلد مخصوص آن در گره فیزیک تغییر می کند.

physics : این فیلد برای قرار تعریف گره فیزیک به منظور مدلسازی فیزیکی و استفاده از خواص فیزیکی است. مانند هل دادن یک توپ توسط ربات ، در این مثال هم ربات و هم توپ باید گره فیزیک داشته باشند.

locked : اگر این فیلد مقدارش TRUE باشد ، جسم صلب دیگر بوسیله موس قابل حرکت نیست (قفل می شود). این فیلد برای جلوگیری از حرکت دادن اشتباهی یک جسم مفید است . اگر FALSE باشد جسم قفل نیست.

۴-۳- گره فیزیک Physics

گره فیزیک به کاربر اجازه تعریف خصوصیات فیزیکی را می دهد که بوسیله موتور شبیه سازی مورد استفاده قرار می گیرد. این گره در بیشتر جهان های مجازی بجز آنهایی که مدل سینماتیکی شده اند مورد استفاده قرار می گیرد.

جدول فیلدها

دامنه فیلد	مقدار پیش فرض	نام فیلد	نوع متغیر
(kg/m^3) اگر ۱- باشد غیر فعال می شود	۱۰۰۰	Density	SFFloat
(kg) اگر چگالی فعال باشد ، جرم نادیده گرفته می شود	-۱	Mass	SFFloat
9 float values	[ ]	inertiaMartix	MFFloat
[0,1]	۰.۰۵	Bounce	SFFloat
(m/s)	۰.۰۱	bounceVelocity	SFFloat
در راهنمای ODE	۱	Coulombfriction	SFFloat
در راهنمای ODE	۰	ForceDependentSlip	SFFloat
)∞,∞(-	۰۰۰	centerOfMass	SFVec3f
وابسته به ماتریس اینرسی	۰۱۰۰	Orientation	SFRotation

تعریف فیلدها

Density, mass : فیلد جرم و چگالی ، جرم کلی ربات را تعریف می کند ، اگر چگالی هر مقداری مثبتی داشته باشد ، جرم نادیده فرض می شود ولی اگر چگالی عدد منهای یک را گرفت آنگاه چگالی نادیده فرض می شود و جرم باید نوشته شود (اولویت با چگالیست).

هیچ گاه هر دو نباید همزمان ۱- باشند و توصیه می شود یکی حتما۱- قرار داده شود. اگر چگالی مقدار مثبت داشته باشد جرم ربات از حاصلضرب چگالی در حجم شی مرزی که در قسمت مرز تعیین می شود بدست می آید ولی این جرم محاسبه شده در فیلد mass نشان داده نخواهد شد.

bounce : این پارامتر ضریب جهش یا ضریب الاستیک یک جسم را نشان می دهد و همواره مقدارش بین صفر تا ۱ است.

ضریب صفر یعنی اینکه جسم هیچ فنریت و جهشی ندارد و عدد یک ماکزیمم جهش هست. در مکانیک برخورد هنگامی که دو جسم صلب با هم بر خورد پیدا می کنند و ضربه می زنند، جهندگی کل ، میانگین ضریب جهش هر دو است و در واقع وابسته به سرعتهای قبل از بر خورد و بعد از بر خورد هر دو شی است.

اگر یکی از اجسام گره فیزیک و طبعا ضریب جهش نداشت ، برای شبیه سازی بر خورد از ضریب جهش دیگر جسم استفاده می شود. این قاعده برای سرعت جهش ، اصطکاک استاتیکی و اصطکاک جنبش نیز استفاده می شود.

force Dependent Slip : این فیلد مقدار ضریب را تعریف می کند که باعث لغزش جانبی جسم خواهد شد. مثلا اگر ماشین ایستاده را عمود بر جهتش به صورت جانبی هل دهید ، حرکت نمی کند ولی ماشین در حال حرکت ، در اثر نیروی جانبی ، به پهلو حرکت خواهد و این به دلیل داشتن ضریب لغزش جانبی است.

inertia Matrix : این فیلد ماتریس اینرسی را تعریف می کند. اگر این فیلد خالی باشد یا دقیقا ۹مقدار عددی نداشته باشد ، نادیده گرفته میشود. علاوه بر این اگر جمر ۱- باشد فیلد ماتریس اینرسی نادیده گرفته می شود.

اگر این فیلد دقیقا ۹ مولفه داشته باشد و فیلد جرم ۱- نباشد آنگاه این ۹ مولفه بوسیله دستور dMassSetParameters() که یک دستور موتور ODE است استفاده می شود. ۹ مولفه این ماتریس به ترتیب چنین هستند : cgx , cgy, (cgx, cgy,cgz) – cgz , I11 , I22 , I33, I12 , I13 , I23 موقعیت مرکز گرانش در دستگاه بدنه ربات است و Iها همان اینرسی جرمی ربات یا مولفه های ماتریس اینرسی هستند که بدین صورت نوشته می شوند . واحد مولفه های ماتریس $kg.m^2$ است.

$[I_{11} \quad I_{12} \quad I_{13}]$

$[I_{12} \quad I_{22} \quad I_{23}]$

$[I_{13} \quad I_{23} \quad I_{33}]$

Center Of Mass : این فیلد موقعیت مرکز جرم صلب را مشخص می کند که بر حسب متر است و نسبت به دستگاه مختصات گره صلب تعریف می شود.

۴-۴- گره تبدیل دستگاه مختصات Transform

گره تبدیل یا تغییر در بسیار از گره ها وجود دارد و همچنین خود نیز به عنوان یک گره مستقل نیز هست که یک سیستم مختصات را به عنوان مبدای برای گره های واقع در فیلد children خودش ، ایجاد می کند.

این دستگاه موقعیت و دورانش در فیلدهای به همین نام نسبت به گره والدینش تعیین می شود یعنی دستگاه واسطه ای بین بچه ها و والدینش (منظور فقط یک گره بالا دست است) است.

جدول فیلدها

دامنه فیلد	مقدار پیش فرض	نام فیلد	نوع متغیر
(-∞,∞)	0,0,0	Translation	SFVec3f
[-1,1], (-∞,∞)	0,1,0,0	Rotation	SFRotation
(-∞,∞)	1,1,1	Scale	SFVec3f

تعریف فیلدها

- **Translation :**

این فیلد بدار انتقال بین دستگاه مختصات چسبیده به گره والدین و دستگاه مختصات چسبیده به گره های بچه ها را نشان می دهد.

- **Rotation :**

این فیلد محور دوران و مقدار دوران مانند بردار دوران اویلر برای دستگاه مختصات چسبیده به children نسبت به گره والدین ایجاد می کند. این فیلد شامل چهار مقدار اعشاری است.

Alpha, r_x , r_y , r_z که سه مولفه اول محوط دوران نرمال شده (بردار یکه بدون واحد ، فقط جهت) که دوران حول آن انجام می گیرد را نشان می دهد و مولفه چهارم مقدار دوران به رادیان است . همه مقادیر می توانند مثبت یا منفی باشند.

توجه داشته باشید که حتما طول محور $r_xi+r_yj+r_zk$ برابر یک باشد یعنی مجموع مربعات این سه مولفه همواره برابر عدد یک باشد $\left(r_x^2+r_y^2+r_z^2=1\right)$در غیر این صورت شبیه سازی غیر قابل تعریف است. در حقیقت این سه مولفه کسینوسهای هادی سه برادر هستند:

$$r_x=\cos(\theta_x)$$

$$r_y=\cos(\theta_y)$$

$$r_z=\cos(\theta_z)$$

که θ_i زاویه محور دوران نسبت به محور i است.

مثال : یک دوران به اندازه $\pi/2$ رادیان حول محور z چنین می شود:

Rotation 0 , 0 , 1 , 1.5708

دوران به اندازه π رادیان حول بردار نیم ساز محورهای x , y چنین می شود:

Rotation 0.7071, 0.7071, 0, 3.1416

و توجه داشته باشید بعضی مقادیر دوران نتیجه یکسانی دارند مانند :

Rotation 0, 1, 0, -1.5708

Rotation 0, -1, 0, 1.5708

- **scale :** این فیلد در صورت امکان یک مقیاس و تجانس غیر یکنواخت ایجاد می کند. تغییر مقیاس فقط برای اشیای گرافیکی امکان پذیر است (غیر مرزی و غیر صلب) .

فیلد مقیاس در تمام گره های صلب و آنهایی که صلب محسوب می شوند و اشیا مرزی همواره باید ۱ و ۱ و ۱ باقی بماند چون موتور شبیه سازی تغییر مقیاس اجسام صلب را پشتیبانی نمی کند.

۴-۵- گره ربات Robot

ربات یک گره ریشه ای برای ساختن ربات مجازی است. فیلدهای این گره در گره های Supervisor , Differential Wheels نیز وجود دارد ، چون این دو گره نیز نقش ربات را دارند.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	دامنه فیلد
SFSrting	Controller	"void"	
SFSrting	Controller Args	" "	
SFBool	Synchronization	TRUE	
MFFloat	Battery	[]	
SFFloat	cpuConsumption	0	(0, ∞)
SFBool	SelfCollision	FALSE	

تعریف فیلدها:

- **controller :**

در این فیلد نام برنامه کنترلی که وبتز برای کنترل ربات استفاده می کند آورده می شود. این برنامه در زیر پوشه ی یک پوشه اصلی با نام کنترلر قرار می گرید. مقدار این فیلد باید انتخاب شود نمی توان نام را تایپ کرد.

برای مثال : اگر مقدار فیلد "my controller" باشد ، برنامه کنترلی باید در مسیر زیر باشد:

My project/controllers/my controller/my controller [.exe]

که نوع اجرایی فقط مربوط به سیستم عامل ویندوز است.

- **controller Args :**

مقدار رشته ای شامل مولفه هایی (که بوسیله دکمه فاصله صفحه کلید از هم جدا می شوند) . برای گذاشتن در تابع اصلی ()main برنامه کنترلی به زبانهای c ,c++, java

- **synchronization :**

اگر این فیلد مقدار True یا صحیح را بگیرد که این پیش فرض نیز می باشد ، شبیه ساز و برنامه کنترلی همزمان اجرا می شوند و اگر مقدار FALSE را به آن بدهید ، شبیه ساز در سریع ترین حالت خود اجرا می شود بدون اینکه همزمانی داشته باشد.

دستور ()wb_robot_get_synchronization برای خواندن مقدار این فیلد از طریق برنامه کنترلی است.

- **battery :**

بعد از انتخاب این فیلد دکمه insert after را ۳ بار کلیک کنید. این فیلد دارای سه مقدار است اولین مقدار نشانگر مقدار انرژی ربات در حالت حاضر بر حسب ژول است (J) .

مقدار دوم نشانگر ماکزیمم انرژی است که ربات می تواند در خود نگه دارد ، سومین مقدار سرعت دوبارهشارز کردن باطری ربات بر حسب وات است ([w]=[J]/[s]) که می توان برای دوباره شارز از گره شارز به طو جداگانه استفاده کرد . وبتز مقدار اول را در طول حرکت تغییر می دهد.

ولی دو مقدار دیگر ثابت هستند. زمانی ربات انرژی مصرف می کند که فیلدهای بعدی یعنی مصرف پردازشگر ربات و مصرف موتورهای ربات بر حسب وات صفر نباشند و دارای مقدار باشند. این دو فیلد به طور پیش فرض صفر هستند.

نکته مهم : هنگامی که مقدار انرژی سطح اول به صفر برسد ، حرکت ربات متوقف می شود و شبیه سازی تمام می شود. در پوشه وسایل نیز مثالی برای آن وجود دارد. دستور

() Wb_robot_battery_get_value مقدار انرژی لحظه ای ربات را به برنامه کنترلی بر می گرداند که البته قبل از استفاده از آن باید سنسور باطری فعال باشد.

- **cpu Consumption :**

مصرف انرژی پردازشگر ربات را بر حسب وات معین می کند.

- **self Collision :**

این فیلد هنگامی که TRUE باشد بر خورد اجزای ربات با هم فعال می شود مانند . بر خورد لینکهای بازوهای رباتی با یکدیگر ، فعال کردن این فیلد احتمالا سرعت شبیه سازی را کاهش خواهد داد.

۴-۶- گره سنسور فاصله یاب Distance Sensor :

گره فاصله یاب برای مدل کردن سنسورهای مادون قرمز ، سونار ، لیزر یا فاصله یاب است. این وسیله برای تشخیص نزدیک شدن به بر خورد مناسب است. بدین صورت که چندین سنسور فاصله موانع اطراف ربات تا ربات را گزارش می دهند. پرتو این سنسور با تیک زدن فیلد Display sensor rays از منوی Tools>Proferences>Rendering قابل مشاهده است. اشعه قرمز قبل بر خورد و اشعه سبز بعد از بر خورد را نشان می دهد.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	دامنه فیلد
MFVec3f	Lookup Table	[0,0,0] [0.2,1000,0]	
SFString	Type	"infra-red"	$(0, \infty)$
SFInt32	numberOfRays	0	$(1, \infty)$
SFFloat	Aperture	1	$(0, 2\pi)$
SFFloat	Gaussian Width	0	

تعریف فیلدها:

- **type :** نوع سنسور را مشخص می کند که می تواند این کلمات باشد:

"infra-red" ، که مقدار پیش فرض است. "sonar" ,"Laser". سنسور فاصله یاب با نوع مادون قرمز حساس به اشیا تاریک و غیر قرمز دارد. سنسور فاصله یاب از نوع صوتی یا Sonar و لیزر نسبت به رنگ موانع حساس نیستند.

سنسور از نوع لیزر فقط برای ایجاد پرتوی قرمز لیزری است که روی سطح مانع نقطه ای قرمز رنگ تولید می کند که در دوربین قابل مشاهده است.

- **Lookup Table :**

جدولیست برای مشخص کردن منحنی پاسخ مطلوب و اغتشاش سنسور . این جدول نشان می دهد چطور وبتز پاسخ سنسور را با دستور distance_sensor_get_value() باید ارائه دهد.

ستون اول جدول یعنی مولفه اول هر بردار ، ورودی سنسور یا فاصله از مانع است.

ستون دوم پاسخ مطلوب یا مقدار سطح رنگ قرمز بازتاب شده است و

ستون سوم اغتشاش یا نویز مطلوب را نشان می دهد، مثال

Lookup Table [0, 1000, 0]

[0.1, 1000, 0.1]

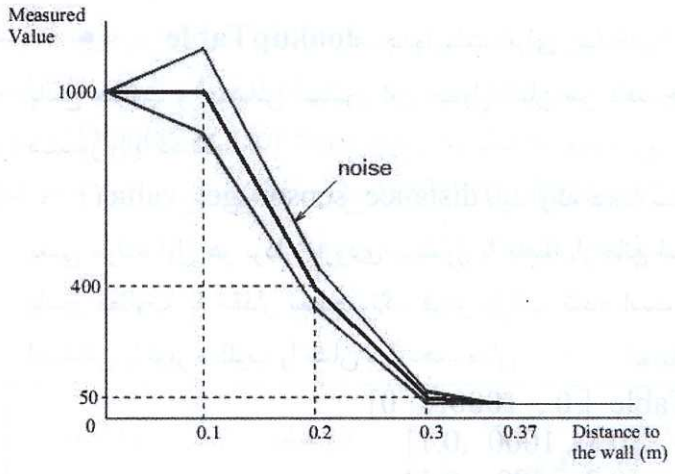
[0.2, 400, 0.1]

[0.3, 50, 0.1]

[0.37, 30, 0]

جدول بالا این معنی را می دهد که در فاصله صفر متر سنسور مقدار ۱۰۰۰ با اغتشاش صفر را بر می گرداند ، در فاصله ۰.۱ متر از دیوار سنسور مقدار ۱۰۰۰ با اغتشاش حداکثر ده درصد را بر می گرداند ، در فاصله ۰.۲ متر ، سنسور مقدار ۴۰۰ را با اغتشاش مثبت و منفی حداکثر ده درصد (یعنی ۴۰) را بر می گرداند و ... نمودار حاصله لزوما خطی نیست .

نمودار شکل ۵-۱ این مفهوم را بهتر می رساند . توجه کنید که ستون اول جدول همواره باید مثبت و صعودی باشد و ستون دوم نزولی . دو خط اطراف نمودار اصلی اغتشاش مثبت و منفی را نشان می دهد.

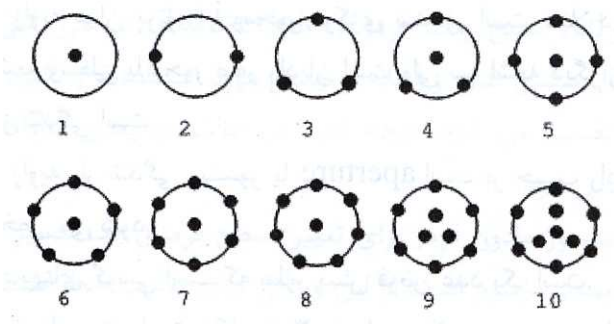


شکل ۵-۱

- **number OfRays :**

تعداد پرتوها خارج شده از یک سنسور را فقط می تواند عدد یک باشد. یک یا بیشتر از آن باشد. اما برای سنسور لیزر فقط می تواند عدد یک باشد . اگر این عدد برای سنسورهای مادون قرمز و سونار بیشتر از یک باشد ، چندین پرتو از سنسور بیرون می آید که مقدار اندازه گیری شده میانگین وزنی همه پرتوهاست.

بوسیله سنسور چند پرتویی مدل کردن فیزیکی سنسور مادون قرمز یا مافوق صوت امکان پذیر می شود. پرتوهای سنسور مانند یک هرم سه بعدی پخش می شود که زاویه باز شدگی آنها در فیلد aperture مشخص می شود. شکل ۵-۲ . حد بالایی برای تعداد پرتو وجود ندارد اما وبتز تعداد بسیار زیاد پرتوها را نادیده می گیرد.



شکل ۵-۲

aperture :

زاویه بین پرتوهای یک سنسور یا شعاع پرتولیزر را مشخص می کند. برای سنسور مادون قرمز و سونار ، این فیلد مقدار باز شدگی اشعه ها را بر حسب رادیان مشخص می کند. هنگامی که چندتایی باشند زاویه هرم اشعه ها را مشخص می کند. برای نوع لیزری این فیلد شعاع نقطه پرتو لیزر را به متر نشان می شخص می کند.

- **Gaussian Width :**

میانگین وزنی اشعه های سنسور را وقتی سنسور چندین پرتویی باشد را نشان می دهد و اصطلاحا سهم گوسی هر پرتو نام دارد. سهم وزنی هر سنسور چنین محاسبه می شود:

$$w_i = \frac{\exp\left(-\left(\frac{t_i}{a.g}\right)^2\right)}{\sum_i^n =1^{w_i}}$$

در این فرمول که سهم گوسی هر پرتو را نشان می دهد .

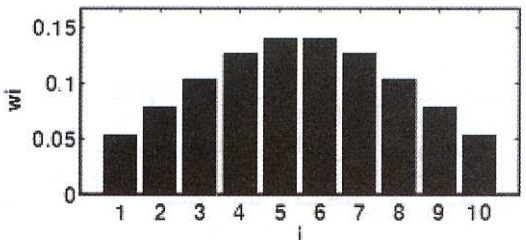
ω_i: سهم وزنی اشعه i است.

t_i : زاویه میان پرتوی i و محور مرکزی سنسور است. مثلا در پرتو چهارتایی زاویه اشعه وسطی با محور صفر رادیان است ولی سه اشعه دیگر زاویه اشان همان زاویه باز شدگی است.

a : زاویه باز شدگی سنسور یا aperture است بر حسب رادیان که در فیلد آن مشخص می شود.

g: پهنای گوسی است که بطور پیش فرض عدد یک است.

n: تعداد پرتوهاست . شکل ۵-۳ پرتوهای مرکزی هرم پرتوهای سنسور سهم وزنی بیشتری نسبت به پرتوهای پیرامون دارد.



سنسور مادون قرمز Infra – Red

در حالت سنسور مادون قرمز ، مقدار بر گردانده شده بوسیله جدول lookup وابسته به ضریب انعکاس است که آن هم به رنگ شی مورد اصابت بستگی دارد. مثلا در مورد یک گره صلب ، رنگ مرز آن برای محاسبه ضریب انعکاس مورد استفاده قرار می گیرد و این با رنگ خود شی تفاوت دارد.

ضریب انعکاس با این فرمول محاسبه می شود:

که red_level سطح رنگ قرمز (diffuse Color) در مرز شی مورد اصابت است. مقدار سنسور قبل از اینکه از جدول مورد استفاده قرار بگیرد بر این ضریب انعکاس تقسیم می شود، توجه کنید در حالت دو بعدی fast2d این ضریب محاسبه نمی شود.

استفاده از سنسور مادون قرمز برای تعیین سطح قرمزی یک تصویر کشید بر روی زمین ، بسیار مرسوم است. از این ویژگی برای تعقیب خط استفاده می شود. این نوع ربات در مثال ربات rover.wbt در مسیرprojects/robots/mindstorms/worlds وجود دارد.

دستورات سنسور فاصله یاب distance sensor

1. Wb distance sensor enable

2. Wb distance sensor disable

3. Wb distance sensor get value

کد دستور به زبان c

#include<webots/distance sensor.h>

Void wb_distance_sensor_enable(**WbDevice Tag**

sensor, **int** ms);

Void wb_distance_sensor_disable(**wbDeviceTeg**

sensor);

Double wb_distance_sensor_get_value

(**wbDeviceTag** sensor);

توصیف دستورات

دستور شماره یک سنسور را فعال می کند هر گام به میلی ثانیه که برایش تعیین کنیم ، در آن گام زمانی با دستور شماره ۳ اطلاعات می گیرد. دستور شماره دو آن را غیر فعال می کند.

دستور ۳ مقادیر سنسور را به یک متغیر بر می گرداند. محدوده این مقادیر در جدول lookup تعیین می شود و بستگی به رنگ مرز شی هم دارد.

۴-۷- گره سرو servo

گره سرو برای افزودن یک درجه آزادی فعال یا غیر فعال بکار می رود. درجه آزادی (حرکت یا چرخش های مستقل از هم) میان گره والدین (سرو) و بچه های سرو یا همان گره children آن بوجود می آید و به اشیایی که در گره children سرو هستند اجازه حرکت خطی یا دوران حول محور سرو را می دهد. سرو دارای دو نوع دورانی یا خطی است که نوع آن در فیلد type آن با نوشتن یکی از دو کلمه rotational یا linear مشخص می شود.

درجه آزادی فعال یعنی سرو به عنوان یک موتور عمل کند و گشتاوری یا نیروی بر اشیای زیر مجموعه اش وارد کند. ماکزیمم این نیرو یا گشتاور در قسمت maxforce معین می شود. درجه آزادی غیر فعال یعنی گشتاور یا نیرویی بر درج آزادی اعمال نشود یا فیلد maxforce برابر صفر باشد.

در این حالت سرو مانند یک مفصل لولایی یا یک مفصل منشوری در حالت خطی عمل خواهد کرد. توجه کنید برای مدل کردن یک مفصل با درجات آزادی بیشتر مانند یونیورسال یا کروی باید نوع سرو را none نوشت که به این معنی است که سرو ، هیچ محدودیتی برای آزادی حرکت بچه هایش قائل نیست و لذا برای مدل کردن مفاصل آزادتر باید از ابزار جانبی فیزیک و کد نویسی با استفاده از موتور ode استفاده کرد. مثال stewart platform در پوشه demo وبتز چنین مفاصلی را مدل کرده است.

گره سرو نوع دورانی برای شبیه سازی مفاصل لولایی و موتورهای الکتریکی و فنرهای پیچشی استفاده می شود.

نوع خطی برای شبیه سازی حرکاتی خطی مانند موتور خطی ، پیستون ، سیلندرهای هیدرودینامیک و پنوماتیک ، فنرهای خطی یا دمپر استفاده می شود.

نوع دورانی بر حسب رادیان گردش می کند و نوع خطی بر حسب متر محاسبه می شود.

جدول ۵-۱ واحد سنجش فیلدهای سرو را نشان می دهد.

فیلد	Rotarional دورانی	Linear خطی
Position	Rad(radians)	M(meters)
Velocity	Rad/s(radians/second)	m/s (meters / second)
Acceleration	Rad/s2(radians/second2)	m/s2 (meters/second2)
Torque/Force	N*m(Newtons*meters)	N(newtons)

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	دامنه فیلد
SFString	Type	"rotational"	
SFFloat	maxVelocity	10	)∞ (0,
SFFloat	Max Force	10	)∞ (0,
SFFloat	controlP	10	)∞ (0,
SFFloat	Acceleration	-1	(0,)∞یا-1
SFFloat	Position	0	
SFFloat	minPosition	0	,0]∞(-
SFFloat	maxPosition	0	)∞[0,
SFFloat	Minstop	0	0,]π[-
SFFloat	maxStop	0	]π[0,
SFFloat	Spring Constant	0	)∞[0,
SFFloat	Damping Constant	0	)∞[0,

تعریف فیلدها:

- **Type :**

یک مقدار رشته ای است که نوع سرو را معین می کند و می تواند مقادیر rotational یا linear باشد که مقدار دورانی پیش فرض است.

- **max Force :**

ماکزیمم نیرو بر حسب نیوتن یا ماکزیمم گشتاور را بر حسب نیوتن – متر برای سرو موتورها معین می کند. دستور servo_set_motor_force_wb() این مقدار را در حین اجرای برنامه از طریق برنامه کنترلی تغییر می دهد.

این مقدار باید صفر یا مثبت باشد و مقدار پیش فرض آن ۱۰ است. اگر مقدار ماکزیمم نیرو کم باشد ، موتور نمی تواند جسم سنگین را بلند کند یا حتی وزن خودش را تحمل کند.

- **Contro IP :**

این فیلد مقدار اولیه پارامتر P را تعیین می کند . پارامتر p بهره تناسبی موتور سرو با کنترل pاست. مقدار بالای عدد p پاسخ های بزرگ و خطاهای کوچک را نتیجه میدهد و بنابراین یک سیستم بسیار حساس خواهیم داشت.

توجه داشته باشید که با دادن مقادیر بسیار بزرگ به p سیستم ناپایدار خواهد شد. با مقادیر p کوچک ، گام های زمانی بسیاری برای رسیدن به موقعیت مطلوب نیاز خواهد بود ، اما سیستم پایداری بیشتری خواهد داشت.

مقدار p در حین اجرا نیز با دستور wb_servo_set_control p() قایل تغییر است.

- **acceleration :**

شتاب پیش فرض سرو موتور را با کنترل p تعیین می کند. مقدار ۱- یعنی اینکه شتاب بوسیله کنترل p محدود نمی شود بلکه نامحدود است. شتاب در حین اجرا نیز بوسیله دستور wb_servo_set_acceleration() قابل تغییر است.

- **Position :**

موقعیت فعلی سرو را نشان می دهد وقتی سرو دورانی است بر حسب رادیان و هنگامی که خطی است بر حسب متر آن را نشان می دهد. هنگامی که شبیه سازی را متوقف می کنید اگر روی این فیلد کلیک کنید موقعیت جاری را نشان می دهد.

- **Max Position , min Position :**

محدوده حد بالا و پایین برای موقعیت سرو یا همان فیلد position ایجاد می کند.

- **Max Stop , min Stop :**

موقعیت توقف فیزیکی یا مکانیکی سرو را مشخص می کنند که در قسمت محدودیت های سرو بیشتر توضیح داده خواهد شد.

- **damping Constant , spring Constant :**

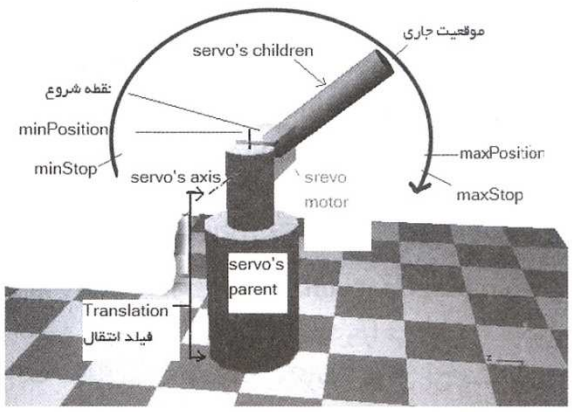
این فیلدها اجازه افزودن فنر یا دمپر (میرا کننده) به سرو را می دهند. همچنین اگر سرو مانند یک مفصل عمل کند یعنی ماکزیمم نیروی آن صفر باشد این فیلد درجه آزادی مورد نظر را به یک فنر پیچشی یا خطی تبدیل می کند.

۴-۷-۱ – موقعیت و جهت گیری اولیه سرو

گره servoفیلدها rotation , translation را مانند گره Transform دارا است. این دو فیلد بیانر موقعیت و جهت گیری دستگاه مختصات چسبیده به وسط محور سرو (متصل به بچه های سرو) نسبت به گره بالا دستش یا والدینش است که والدین آن می توانند یک ربات یا سرو موتور دیگری باشند.

هنگامی که سرو در حال حرکت است ، مقادیر این دو فیلد به طور خودکار اصلاح می شوند تا با وضعیت تبدیل مختصات جدید منطبق شوند.

در نوع سرو دورانی ، فیلد انتقال ، موقعیت وسط محور دوران سرو را نشان می دهد و فیلد rotation جهت گیری محور دوران را نشان میدهد شکل ۵-۴ را ببینید.



شکل ۵-۴

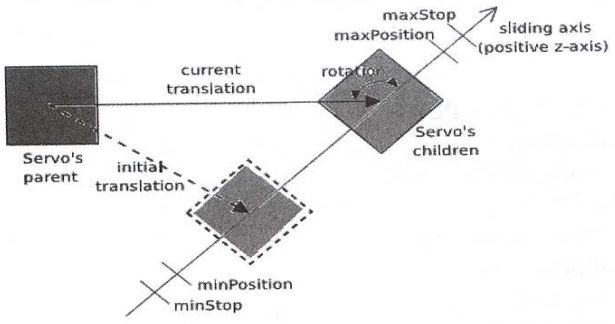
در نوع سرو خطی فیلد انتقال مختصات محور لغزنده را نشان میدهد و فیلد دوران ، جهت گیری محور لغزنده را نشان می دهد. مقادیر فیلدهای انتقال و دوران قبل از اجرای شبیه سازی بیانگر مختصات و جهت گیری اولیه سرو است.

در این هنگام فیلد position صفر است، سپس موقعیت مطلوب توسط برنامه کنترلی و دستور

Wb_servo_set_position() نسبت به موقعیت اولیه تعیین می شود. مقادیر , maxStop minPosition , maxPosition,minStop نسبت به موقعیت صفر اولیه تعیین می شود. شکل ۵-۵

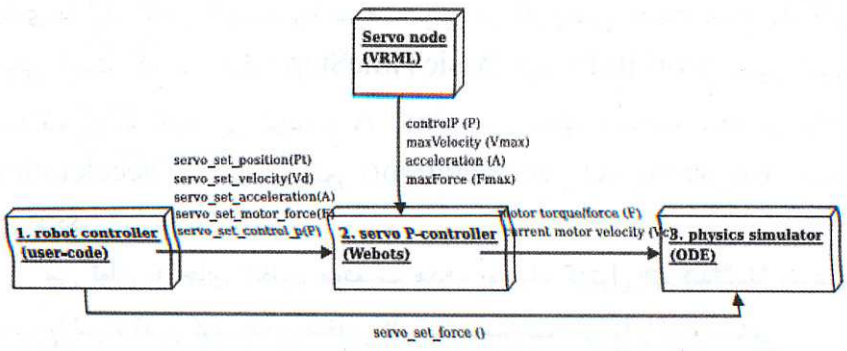
۴-۷-۲- کنترل سرعت و موقعیت

راه متعارف استفاده از سرو ، کنترل مستقیم موقعیت آن است (control Position) . کاربر بوسیله دستور wb_servo_set_position() موقعیت هدف را به سرو می دهد. سپس کنترل p شتاب ، سرعت و نیروی مورد نیاز برای دست یافتن به این موقعیت را محاسبه و سرو را تنظیم می کند. جدول ۵-۲ را ببینید.



شکل ۵-۵

در وبتز ، کنترل موقعیت در سه مرحله انجام می شود که در شکل ۵-۶ کشیده شده است. مرحله اول بوسیله برنامه کنترلی کاربر (شماره۱) . کاربر تصمیم می گیرد چه موقعیت ، سرعت ، شتاب و نیرویی باید اعمال شود. مرحله دوم بوسیله کنترل p(شماره ۲) انجام می گیرد که سرعت جاری را محاسبه می کند (v_c) سر انجام مرحله سوم بوسیله موتور شبیه ساز فیزیکی ODE انجام می شود.



شکل ۵-۶

در هر گام شبیه سازی ، کنترل P ، سرعت جاری را با الگوریتم زیر محاسبه می کند.

```
Vc=P * (Pt –Pc);

If(abs(Vc)>Vd)

Vc=sign(Vc) * Vd;

If (A !=-1) {

a= (Vc – Vp)/ts;

if (abs(a)>A)

a=sign(a) * A;

Vc=Vp+a * ts;
```

}

در این الگوریتم V_c سرعت جاری بر حسب رادیان بر ثانیه یا متر بر ثانیه است. P بهره تناسبی در فیلد controlP مشخص می شود.

p_t موقعیت هدف است که با دستور Wb_servo_set_position() مشخص می شود. pc موقعیت جاری سرو است.

V_d سرعت مطلوب که با فیلد max Velocity تعیین می شود یا با دستور خاص آن توسط برنامه کنترلی.

a شتاب مورد نیاز برای رسیدن به سرعت جاری در یک گام زمانیست .

V_p سرعت موتور در گام زمانی گذشته است.

T_s گام زمانی است که در فیلد basic TimeStep گره worldInfo تعیین می شود (تبدیل به ثانیه می شود) و A شتاب سرو موتور مشخصه شده در فیلد acceleration است یا با دستور wb_servo_set_acceleration() معین می گردد.

می توان به جای تنظیم موقعیت هدف بوسیله کنترل p ، مستقیما سرعت سرو را نیز تنظیم کرد. بدین منظور کاربر باید سرعت سرو را با این دستور wb_servo_set_velocity() تنظیم کند سپس دستور ±wb_servo_set_position(servo_name,WB_SERVO_INFINITY) را وارد کند.

این کار باعث آغاز حرکت پیوست سرو با سرعت مطلوب شبیه یک موتور dc تا بی نهایت می شود.

جدول ۵-۲: سه روش کنترل servo

روش کنترل نیرو	روش کنترل سرعت	روش کنترل موقعیت	دستورات
نه	بله	بله	استفاده P-controller از
-	+/- WB_SERVO_INFINITY	موقعیت مطلوب	...set_Position()
صرف نظر	سرعت مطلوب	سرعت ماکزیمم مطلوب	...set_velociry()
طرف نظر	ماکزیمم شتاب	ماکزیمم شتاب	...set_ acceleration()
ماکزیمم نیرو	ماکزیمم نیرو	ماکزیمم نیرو	...set_motorforce()
نیروی مطلوب	-	-	...set_force()

۴-۷-۳- کنترل نیرو

دو روش کنترل موقعیت و سرعت از کنترلر P استفاده می کردند ولی یک روش سوم نیز وجود دارد، وبتز به کاربران اجازه می دهد تا مستقیما نیرو یا گشتاور مورد نظرشان را بر شی اعمال کنند. این عمل از طریق wb_servo_set_force() انجام می شود.

محدوده های سرو

فیلد position در حالت دورانی نشان دهنده زاویه بین نقطه شروع و موقعیت کنونی به رادیان است و در مورد خطی نشان دهنده فاصله (به متر) میان نقطه شروع و فیلد انتقال کنونی است.

maxPosition , minPosition محدوده نرم سرو را نشان می دهند. محدوده نرم ، مرزی را برای کنترل P مشخص می کند که خارج از آن برای حرکت دادن سرو تلاشی نکند. اگر بوسیله فرامین موقعیت هدف بیرون از این محدوده تعریف شود.

آن موقعیت در محدوده نرم قطع می شود. از آنجایی که موقعیت اولیه سرو همیشه صفر است . فیلد minPsition باید صفر یا منفی باشد و ماکزیمم موقعیت باید همواره مثبت یا صفر باشد. هنگامی که هر دوی این فیلدها صفر باشند (پیش فرض) محدوده نرم غیر فعال می شود.

توجه کنید که محدوده نرم می تواند از محدوده خودش بیشتر شود مثال هنگامی که وزن ربات از نیروی موتوری لازم برای بلند کردن ربات ، بیشتر باشد.

فیلدها minStop , maxStop محدوده سخت سرو را مشخص می کند. این محدوده سخت ، محدوده فیزیکی است که بوسیله هیچ نیرویی نمی تواند نقض شود. م حدوده سخت برای شبیه سازی پیستونهای هیدرولیک یا پنوماتیک یا یک محدوده بسته مفصل لولایی استفاده می شود.

مقدار minStop باید در بازه بسته [-pi,0] باشد و maxStop در بازه بسته [pi,0] باید باشد. وقتی هر دو صفرند (پیش فرض) محدوده سخت غیر فعال است. به یاد داشته باشید که هنگامی که هر دو محدوده نرم و سخت ، فعال باشند. محدوده نرم باید درون محدوده سخت باشد.

۴-۷-۴- فنرها و میراکننده ها

فیلد spring Constant ثابت سختی فنر را معین می کند. معمولا با حرف k نشان داده می شود. این فیلد باید دارای مقدار صفر یا مثبتی باشد. اگر صفر در نظر گرفته شود (پیش فرض) هیچ نیرو یا گشتاور فنری اعمال نمی شود.

اگر غیر صفر باشد نیروی فنری غیر از سایر نیروها بر سرو وارد می شود (نیروی موتور ، میرا کننده و وزن) . نیروی فنر از قانون هوک با این فرمول بدست می آید F Xx=-K ثابت فنر و x موقعیت فعلی سرو نسبت به موقعیت اولیه است که در فیلد position مشخص است. این نیرو موقعیت سرو را می خواهد به حالت اولیه بر گرداند.

فیلدdamping Constant ثابت میرایی سرو را مشخص می کند. این ثابت باید صفر یا مثبت باشد. اگر صفر باشد (پیش فرض) هیچ نیروی میرا کننده ارتعاشات فنر اعمال نمی شود.

اگر این فیلد بیشتر از صفر باشد. یک نیرو میرا کننده علاوه بر سایر نیروها بر سرو اعمال می شود. نیرو یا گشتاور میرایی متناسب با سرعت سرو است : $F=-B\times v$ که B ثابت میرایی است و V سرعت موثر سرو است که بوسیله موتور فیزیکی محاسبه می شود.

برای شبیه سازی یک فنر یا یک دمپر به تنهایی باید نیرو سرو را عدد صفر قرار داد تا سرو مانند یک مفصل یک درجه آزادی با یک فنر یا دمپر شود و این کار یا بوسیله دستور

wb_servo_set_motor_force (servo,0) انجام می گیرد یا فیلد maxForce را به صفر باید تغییر داد.

۴-۷-۵- ترکیب نیروها

اگر چه سه نوع مختلف نیرو بر سرو وارد می شود (نیروی موتور ، فنر و میرایی) اما همه آنها موازی و در یک راستا هستند و مستقلا می توانند فعال یا غیر فعال شوند.

جدول ۵-۳ را بینید. در هر گام زمانی شبیه سازی نیروها مجددا محاسبه می شوند و بسته به خطی یا دورانی بودن به صورت نیرو یا گشتاور اعمال می شوند.

اعمال نیرو هم بر بچه های سرو و هم به والدینش اتفاق می افتد با مقدار مساوی و در یک راستا ولی با جهت های عکس طبق قانون سوم نیوتن

جدول ۵-۳: ترکیب نیروها

نیرو	پیش فرض	هنگامی فعال می شود که :	هنگامی غیر فعال می شود که :
نیروی موتور	فعال	Servo set position() servo set force()	Servo_set_motor_force(…,0) maxForce==0
نیروی فنر	غیر فعال	SpringConstant>0	Spring constant==0

نیروی میراکننده	غیر فعال	dampingConstant>0	Damping Constant==0

۴-۷-۶- دستورات سرو servo

1. wb_servo_set_Position

2. wb_servo_set_velocity

3. wb_servo_set_acceleration

4. wb_servo_set_motor_force

5. wb_servo_set_control_P

6. wb_servo_enable_position

7. wb_servo_disable_position

8. wb_servo_get_position

9. wb_servo_enable_motor_force_feedback

10. wb_servo_get_motor_force_feedback

11. wb_servo_disable_motor_force_feedback

12. wb_servo_set_force

کد دستورات به زبان c

```
#include<webots/servo.h>

1: void wb_servo_set_position (WbDevice Tag

Servo,double position);

2: void wb_servo_set_velocity (WbDevice Tag
```

Servo,**double** vel);

3: **void** wb_servo_set_acceleration (**WbDevice Tag**

Servo,**double** acc);

4: **void** wb_servo_setmotor_force (**WbDevice Tag**

Servo,**double** force);

5: **void** wb_servo_set_control_p (**WbDevice Tag**

Servo,**double** p);

6: **void** wb_servo_enable_position (**WbDevice Tag** Servo,**int**

ms);

7: **void** servo_disable_position (**WbDevice Tag** Servo);

8: **double** servo_get_position (**WbDevice Tag** Servo);

9. **void** wb_servo_enable_motor_force_feedback(**WbDevice Tag** Servo,**int** ms);

10: **void** wb_servo_disable_motor_force_feedback (**WbDevice Tag** Servo);

11:**double** wb_servo_get_moto_force_feedback (**WbDevice Tag** Servo);

12. void wb_servo_set_force(**WbDevice Tag**

Servo,**double** force);

توصیف دستورات

دستور شماره یک ، موقعیت جدید را برای کنترل p مشخص می کند تا آن با سرعت جاری و نیروی جاری به آن برسد . توجه داشته باشید که این موقعیت نباید بوسیله محدوده های نرم یا سخت منع شده باشد.

مقادیر WB_SERVO_INFINITY مثبت و منفی آن یک دوران یا حرکت خطی بی نهایت را ایجاد می کند. این حرکت بی نهایت بوسیله دستور مجدد تنظیم موقعیت با مقدار جدید یا دستور wb_servo_set_force() می تواند قطع شود.

دستور شماره ۲ سرعت لازم برای رسیدن به موقعیت هدف را مشخص می کند. این سرعت نباید بیشتر از مقدار فیلد max Velocity باشد.

دستور شماره ۳ شتاب لازم برای رسیدن به سرعت ویژه را برای کنترل p مشخص می کند. توجه کنید که شتاب بی نهایت با قرار دادن مقدار ۱- در این دستور تعریف می شود.

دستور ۴ نیرو یا گشتاور لازم برای حرکت را معین می کند. این مقدار نباید از مقدار فیلد maxForce بیشتر باشد.

دستور شماره ۵ مقدار پارامتر p را تغییر می دهد. p پارامتری است که برای محاسبه سرعت جاری V_c از موقعیت جاری و موقعیت هدف p_t مانند:

$v_c=p\times(p_t-p_c)$. اگر p دارای مقدار کوچکی باشد زمان زیادی برای رسیدن به هدف لازم می شود و اگر p خیلی بزرگ باشد سیستم ناپایدار خواهد شد . مقدار پیش فرض p در فیلد control زیر مجموعه گره سرو مشخص می شود.

دستور شماره ۶ اندازه گیری موقعیت یک سرو خاص را فعال می کند. بدین صورت که هر n میلی ثانیه که برای آن در دستور n مشخص می شود ، موقعیت سر توسط دستور ۸ گرفته می شود. دستور شماره هفت این وضعیت را غیر فعال می کند.

دستور شماره ۸ موقعیت موثر سرویی را که قبلا فعال شده باشد را گرفته و در یک متغیر قرار می دهد. موقعیت موثر یعنی موقعیتی که تحت اثر نیروهای خارجی نیز هست و با موقعیت هدف تفاوت دارد. برای سرو های دورانی مقدار باز گردانده شده به رادیان است و برای سرو خطی به متراست.

دستور شماره ۹ این دستور مانند دستور ۶ است به جز اینکه اندازه گیری نیرو یا گشتاور را برای یک سرو خاص هر n میلی ثانیه فعال می کند و اصطلاحا فیدبک یا باز خورد می کند. مقدار نیرو یا گشتاور توسط دستور ۱۱ گرفته خواهد شد.

دستور ۱۱ نیرو یا گشتاور را که اکنون سرو برای حرکت مورد استفاده قرار می دهد به یک متغیر اعشاری باز می گرداند. در سرو دورانی واحد این متغیر نیوتن – متر [N*M] و در حالت خطی نیوتن است. طبیعتا مقدار باز گردانده شده از مقدار فیلد maxForce یا مقدار مشخص شده در تنظیم نیرو بیشتر نخواهد بود.

نکته: دستور یازده فقط نیروی موثر موتور را اندازه می گیرد و همه نیروهای داخلی و خارجی وارد شده بر سرو را نادیده می گیرد. این دستور نیروهای زیر را اندازه نمی گیرد:

- نیروهای فنر و میرا کننده وقتی که فیلد آنها صفر نباشد.
- نیرو معین شده با دستور wb_servo_set_force()
- نیروهای قیدی که حرکت سرو را مقید به یک درجه آزادی می کنند.

به معنی دیگر فقط نیروهای عمل کننده در درجه آزادی سرو محسوب می شوند نه نیروها و گشتاورهای عمود یا زاویه دار با آن . به عنوان مثال در سرو خطی ، نیروی وارد شده با زاویه ۹۰ درجه نسبت به محور لغزنده به هیچ وجه گزارش داده نمی شود و در سرو دورانی فقط گشتاورهای حول محور دوران در نظر گرفته می شوند.

توجه کنید که این دستور فقط برای مدلسازی فیزیکی است و طبیعتا باید مرز و فیزیک برای گره سرو تعریف شده باشد.

دستور ۱۲ را نباید با دستور شماره ۴ که نیروی موتور را تنظیم می کرد اشتباه گرفت. این دستور به کاربر اجازه می دهد تا مستقیما نیروی که باید سرو اعمال کند را مشخص کند. این دستور در واقع کنترل p و موتور مفصل ODEرا دور می زند.

این دستور مستقیما نیروی را به شبیه سازی فیزیکی وارد می کند و به کار اجازه طراحی یک کنترل کننده دلخواه را می دهد. برای مثال یک کنترل کننده PID (تناسبی – انتگرالی-مشتقی).

مقدار مثبت نیرو یا گشتاور ، بچه های سرو را در جهت مثبت حرکت می دهد (جهت افزایش فیلد Position) این نیرو نباید بیشتر از مقدار فیلد maxForce یا مقدار تعیین شده در دستور wb_servo_set_motor_forc() باشد . برای اعمال این دستور باید مرز و فیزیک برای سرو معین شده باشد.

با دستور ۱۲ همچنین می توان یک فنر یا میرا کننده نیز طراحی کرد. برای دیدن مثالی از این روش به مسیر projects/samples/howto/worlds/force control.wbt رفته و مثال کنترل نیرو را ببیند.

۴-۸- گره سنسور GPS

این گره برای مدل کردن سیستم موقعیت یاب جهانی GPS استفاده می شود که این سنسور موقعیت مطلق را در هر لحظه با استفاده از برنامه کنترلی در دستگاه مختصات دکارتی ارائه می دهد.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض
SFString	Type	"satellite
SFFIoat	Resolution	۰.۰۰۱

تعریف فیلدها

- **type:** این فیلد نوع تکنولوژی GPSرا مشخص می کند. شامل "satellite" یا "laser" است که لیزر فعلا نادیده گرفته می شود.
- **resolution :** این فیلد دقت سنسور را تعریف می کند که در واقع ماکزیمم خطای آن در موقعیت مطلق است (بر حسب متر

دستورات سنسور GPS

1. wb_gps_enable

2. wb_gps_disable

3. wb_compass_get_values

کد دستور به زبان c

#include<webots/gps.h>

Void wb_gps_enable (**wbDeviceTag** gps,int ms);

Void wb_gps_disable (**wbDeviceTag**gps,gps);

const double * wb_gps_get_values (**wbDeviceTag** sensor);

توصیف دستورات

دستور شماره یک سنسور GPS را برای اندازه گیری موقعیت هر میلی ثانیه فعال می کند و دستور شماره دو آن را غیر فعال می کند.

دستور شماره ۳ مقادیر اندازه گیری شده کنونی را به برنامه باز می گرداند. این مقادیر در یک بردار سه بعدی قرار می گیرند. بنابراین فقط مقادیر ۰ و ۱ یا ۲ می توانند به عنوان اندیس بردار باشند ، که اندیس صفر مولفه x ، اندیس ۱ مولفه y و اندیس ۲ مولفه z (مولفه سوم) بردار است. مثلا

wbDevice Tag gps=wb_robot_get_device("gps");

wb_gps_enable (gps,**TIME_STEP**);

const double *gps_values=

wb_gps_get_values("gps");

آنگاه با این دستورات مقدار بر گردانده شده بدین صورت است.

مولفه اول بردار موقعیت gps_values[0] =

مولفه دوم بردار موقعیت gps_values[1] =

مولفه سوم بردار موقعیت gps_values[2] =

مثال مربوط به این سنسور را در پوشه project\sample\device\gps می توانید ببنید.

۴-۹- گره سنسور قطب نما compass

گره قطب نما برای مدل کردن قطب نمای دیجیتالی یک محوره ، دو محوره یا سه محوره به استفاده می شود. گره قطب نما یک بردار طولی واحد که نشان دهنده جهت شمال مجازی است را ارائه می دهد. شمال مجازی بوسیله فیلد northDirection در گره WorldInfo مشخص می شود. توجه کنید که قطب نمای معمولی فقط زاویه نسبت به یک محور را نشان می دهد مثلا فقط نسبت به محور x یا شمال یا زاویه افقی Yaw(azimuth) . قطب نما با دو محور ، زاویه تراز یا زاویه عمودی نسبت افق Pitch را نیز نشان می دهد (Elevation). زاویه سوم همان زاویه Roll در زوایای اویلر است . شکل ۵-۶

شکل ۵-۶

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	توضیحات
MFVec3f	lookupTable	[ ]	درون یابی
SFBool	xAxis	TRUE	محاسبه محور x
SFBool	xAxis	TRUE	محاسبه محور y
SFBool	xAxis	TRUE	محاسبه محور z

تعریف فیلدها

- **lookupTable :** این فیلد اختیاری است . این فیلد یک جدولی را تهیه می کند که مولفه های بردار (بین ۱- تا ۱) در آن قرار داده می شود. با این جدول می توان اغتشاش را به سیستم اضافه کرد و مقادیر ماکزیمم و مینیمم را تعریف کرد. به طور پیش فرض این جدول خالی است و بنابراین مولفه های بردار با طول واحد خروجی قطب نما خواهند بود.
- **xAxis, yAxis , zAxis :** هر کدام از این فیلدها مشخص می کند که محاسبات در راستای آن محور محاسبه نمی شود و دستور ویژه آن مولفه نیز همواره مقدار صفر را خواهد داد. در حالت پیش فرض هر سه TRUE یا فعال هستند.

دستورات قطب نما

1. wb compass enable

2. wb compass disable

3. wb compass get values

کد نویسی دستورات به زبان c

#include<webots/compass.h>

1: **void** wb_ compass _enable (**WbDevice Tag** Sensor, **int**

ms);

2: **void** wb_ compass _disable (**WbDevice Tag** Sensor);

3: **const double** * wb_ compass _get_values

(**WbDevice Tag** sensor);

توصیف دستورات

دستور شماره یک قطب نما را برای اندازه گیری هر میلی ثانیه ای فعال می کند. البته می تواند ، این اندازه گیری را هر چند میلی ثانیه که لازم باشد انجام داد و باید به جای میلی ثانیه ، عدد آن را قرار داد. دستور شماره ۲ قطب نما را غیر فعال می کند.

دستور شماره ۳، یک بردار سه بعدی به برنامه بر می گرداند که اندیس دسترسی به مولفه های این بردار مطابق قبل می تواند ۱ ، ۰ یا ۲ باشد. بزرگی هر کدام از مولفه ها واحد می باشد ، مگر اینکه جدول lookup برداری را که نشان دهنده جهت شمال مجازی نسبت به وضعیت کنونی ربات است را مشخص کرده باشد.

مثال : در دستگاه جهانی وبتز ، صفحه xz نماینده افق است و محور y نماینده ارتفاع است. بردار پیش فرض برای جهت شمال که در فیلد morthDirection مشخص می شود برابر [1,0,0] است که با جهت مثبت محور x مطابقت دارد.

اگر قطب نما موازی افق باشد و بر محور y عمود باشد ، زاویه چرخش ربات بر حسب درجه چنین بدست می آید:

```
Double get_yaw_in_degrees(){
Const double *north= wb_compass_get_values(tag);
Double rad=atan2(north[0],north[2]);
Double yaw=(rad- 1.5708)/M_PI*180.0;
If(yaw<0.0)
Yaw=yaw + 360.0;
Return yaw;
}
```

در این برنامه متغیر rad زاویه ربات یا جهتش را نسبت به شمال را بر حسب رادیان و زاویه yaw جهت ربات را نسبت به شمال به درجه بر می گرداند. متغیر M_PI همان عدد π است و north همان بردار

سه بعدی است که مولفه اول آن برابر سینوس و مولفه سوم آن برابر کسینوس زاویه افقی نسبت به شمال است.

۴-۱۰- گره سنسور ژیروسکوپ Gyroscope

گره ژیرو یا دوران نما برای مدل کردن سنسور سرعت زاویه ای در یک ، دو یا سه محور استفاده می شود. سرعت زاویه ای بر حسب رادیان بر ثانیه اندازه گیری می شود.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	توضیحات
MFVec3f	lookupTable	[ ]	درون یابی
SFBool	xAxis	TRUE	محاسبه محور x
SFBool	xAxis	TRUE	محاسبه محور y
SFBool	xAxis	TRUE	محاسبه محور z

تعریف فیلدها

- **lookup Table :**

این فیلد اختیاری تهیه یک جدول و نگاشتن مقادیر سرعت زاویه در آن است . با این جدول امکان افزودن اغتشاش و تعریف مقادیر ماکزیمم و مینیمم خروجی وجود دارد. به طور پیش فرض این جدول خالیست.

- **xAxis , y Axis , zAxis:**

هر کدام از این فیلدها مشخص می کند که محاسبات در راستای آن محور فعال باشد یا نباشد. اگر هر یک از اینها FALSE باشد ، مولفه مربوط به آن محور محاسبه نمی شود و دستور ویژه آن مولفه نیز همواره مقدار صفر را خواهد داد. در حالت پیش فرض هر سه TRUE یا فعال هستند.

دستور ژیرو

1. wb_gyro_enable

2. wb_gyro_disable

3. wb_gyro_get_values

کد نویسی دستورات به زبان c

#include<webots/gyro.h>

1: **void** wb_gyro_enable **(WbDeviceTeg** sensor,int ms);

2: **void** wb_gyro_disable **(WbDeviceTeg** sensor);

3.**const double** ***wb_gyro_get_values **(WbDeviceTeg** sensor);

توصیف دستورات

دستور اول اندازه گیری سرعت زاویه ای هر میلی ثانیه را فعال می کند.

دستور دوم سنسور ژیرو را غیر فعال می کند.

دستور سوم مقادیر اندازه گیری شده توسط ژیرو را در حال حاضر به برنامه بر می گرداند. مقدار بر گردانده شده یک بردار سه مولفه ایست که برای دسترسی به مولفه هایش فقط مقادیر ۰ ، ۱ و ۲ برای اندیس قابل استفاده اند. هر مولفه بیانگر سرعت زاویه ای حول یک محور ژیروست که بر حسب رادیان بر ثانیه بیان می شود.

مولفه اول بیانگر سرعت زاویه ای حول محور x است.

مولفه دوم حول محور y و

مولفه سوم حول محور z.

توجه کنید که سنسور ژیرو باید جایی در ربات قرار داده شود که در معرض دوران باشد یعنی تمام سرعت زاویه هایی که باید اندازه گیری شوند ، آن را بچرخانند. در پوشه devices زیر مجموعه پوشه

sample در پوشه پروژه های وبتز مثالی را از ژیروسکوپ و همچنین دیگر سنسورهای موقعیت مانند قطب نما و GPS وجود دارد.

۴-۱۱- گره دوربین Comera

گره دوربین برای مدل کردن یک دوربین رنگی یا سیاه و سفید فیلمبرداری متصل به ربات به منظور دیدن تصاویر ، پردازش رنگ و پردازش تصویر است یا در نوع دیگر یک دوربین فاصله یاب (range-finder) است. نوع دوربین در فیلد type آن مشخص می شود. اگر فیلد ارتفاع آن مقدار یک باشد ، دوربین خطی یا یک بعدی می شود.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	توضیحات
SFFloat	fieldOfView	0.7854	۰تاπ
SFInt32	Width	64	واحد : پیکسل
SFInt32	Height	64	واحد : پیکسل
SFString	Type	"color"	"black and white" "range-finder"
SFBool	Display	TRUE	
SFFloat	Near	0.01	واحد : متر
SFFloat	Far	50.0	واحد : متر
SEVec2f	WindowPosition	0 0	۰تا۱
SFInt32	pixelSize	1	
SFBool	antiAliasing	FALSE	
SFInt32	gaussianNoisc	۰.۰	۰تا ۲۵۵

تعریف فیلدها

- **fieldOfView :**

این فیلد زاویه افقی میدان دید دوربین را مشخص می کند. مقدارش بین صفر تا پی رادیان است. چون پیکسل های دوربین مربعی هستند ، میدان دید عمودی از سه فیلد fieldOfView,height,width قابل محاسبه است ، بدین ترتیب

Vertical FOV=fieldOfView *height/width

- **width :** پهنای تصویر بر حسب پیکسل است.
- **height :** ارتفاع تصویر بر حسب پیکسل است.
- **type:** نوع دوربین یکی از این سه مقدار باید باشد: black , color and whit یا range-finder
- **display:** یک پنجره جداگانه برای نمایش تصاویر دریافتی از دوربین باز می کند. اگر مقدار آن TRUE باشد. پنجره تصاویر فعال می شود و اگر FALSE باشد غیر فعال است.
- **far,near:**

فاصله از دوربین تا صفحه گرافیکی نزدیک و دور را مشخص می کنند که در واقع عمق دید دوربین را تعریف می کند. این صفحات با دریچه دوربین موازی هستند. این دو فیلد به همراه فیلد زاویه میدان دید یک دریچه دوربین موازی هستند .

این دو فیلد به همراه فیلد زاویه میدان دید یک هرم سر بریده را تشکیل می دهند که در صورت نیاز قابل مشاهده نیز است (رجوع به بخش ارجحیت ها نمایش) هر شکل گرافیکی خارج از این هرم در تصاویر دوربین دیده نخواهد شد.

بنابراین ، اشکال خیلی دور (دوربتر از فیلد far) دیده نخواهند شد. به طور مشابه اشیای خیلی نزدیک (قبل از فیلد near) نیز توسط دوربین دیده نخواهند شد.

- **window Position :**

موقعیت پنجره تصاویر دوربین را در پنجره اصلی شبیه سازی مشخص می کند. مقادیر x , y در این فیلد اشاره به یک نقطه شناور دارند بین ۰ تا ۱ دارند. این نقطه مرکز پنجره تصاویر را نسبت به گوشه چپ و بالای پنجره اصلی مشخص می کند.

موقعیت صفر و صفر گوشه چپ و بالا و موقعیت ۱و ۱ گوشه راست و پایین را نشان می دهد. هر گاه پنجره اصلی مقیاسش تغییر کند ، پنجره تصاویر دوربین نیز به همان میزان مقیاسش تغییر می کند. همچنین این پنجره را می توان به وسیله موص در پنجره اصلی جابه جا کرد.

هر گاه یکی از دو مقدار این فیلد برابر ۱- قرار داده شود دیگر این پنجره درون پنجره اصلی باز نمی شود بلکه در پنجره ای بیرون و جداگانه نمایش داده می شود و این برای زمانی مفید است که بخواهیم فقط پنجره تصاویر را ببینیم و تصمیم گیری کنیم.

- **pixelsize :**

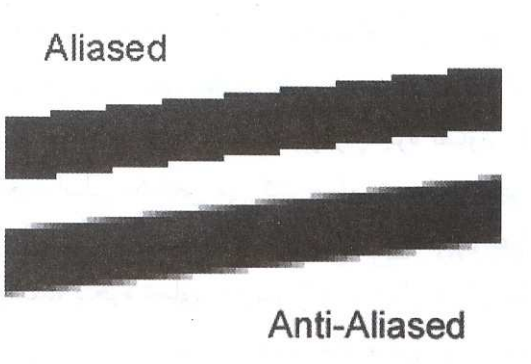
این فیلد سایز پیکسل های نمایش داده شده را نشان می دهد و در حقیقت ضریب بزرگنمایی نمایش است. اگر بزرگتر از یک انتخاب شود تصاویر دوربین درشت تر نشان داده خواهند شد.

- **antiAliasing :**

هر گاه این گزینه TRUE باشد (پیش فرض) فعال است. یکی از مهم ترین تکنیک ها برای آسان تر خوانده شدن متون و ساختن نماهای گرافیکی و جلوه بخش تر شدن سطوح گرافیکی فیلتر Antialiasing است. یک راه موثر و عملی برای واضح تر به نظر رسیدن تصاویر است.

این فیلد کاری می کند که تصاویر در یک صفحه با روزلیشن کم تا حدی صافتر و صیقلی به نظر برسد(شکل۵-۷)

این فیلد تنها تصاویر را نرم تر می کند که شما دندانه های شکسته ی آن را نبینید. Anti aliasing راهیست برای فریب دادن دیدگان شما برای دیدن خطوط صاف و دندانه های نرم در جایی که در حقیقت هیچ چیز نیست.



شکل ۵-۷

- **gaussianNoise :**

اگر این فیلد مقدارش بیشتر از صفر باشد ، ۵ تصاویر دریافتی دوربین یک اغتشاشی معروف به اغتشاش گوسی وارد می کند. دامنه این فیلد بین صفر تا ۲۵۵ است. در این حالت شبیه سازی دوربین واقعی تر خواهد شد.

نکته : در نمایش دوربین در پنجره ای جداگانه و بیرون از پنجره اصلی ، سایه ها و آن چیزی که به وسیله قلم روی زمین رسم می شود، نمایش داده نمی شود.

دستورات دوربین

1. wb_camera_enable

2. wb_camera_disable

3. wb_camera_get_fov

4. wb_camera_set_fov

5. wb_camera_get_width

6. wb_camera_get_height

7. wb_camera_get_near

8. wb_camera_get_far

9. wb_camera_get_rype

10. wb_camera_get_image

11. wb_camera_image_get_red

12. wb_camera_image_get_green

13. wb_camera_image_get_blue

14. wb_camera_image_get_grey

15. wb_camera_get_range_image

16. wb_camera_range_image_get_value

17. wb_camera_move_window

18. wb_camera_save_image

کد نویسی دستورات به زبان C

#include<webots/camera.h>

1: **void** wb_camera_enable(**WbDeviceTag**camera,int

Ms);

2: **void** wb_camera_disable(**WbDeviceTag** camera);

3: **void** wb_camera_get_fov(**WbDeviceTag**

camera);

4: **void** wb_camera_set_fov(**WbDeviceTag**

Camera,**double** fov);

5: **int** wb_camera_get_width(**WbDeviceTag**

camera);

6: **int** wb_camera_get_height(**WbDeviceTag**

camera);

7: **double** wb_camera_get_near(**WbDeviceTag**

camera);

8: **double** wb_camera_get_far(**WbDeviceTag**

camera);

9: **int** wb_camera_get_type();

10: **unsigned ***wb_camera_get_image

(WbDeviceTag camera);

11: **unsigned char** wb_camera_image_get_red

(**const unsigned char** *image,int width,**int** x,int y);

12: **unsigned char** wb_camera_image_get_green

(**const unsigned char** *image,int width,**int** x,int y);

13: **unsigned char** wb_camera_image_get_blue(**const**

unsigned char *image,**int** width,**int** x,**int** y);

14: **unsigned char** wb_camera_image_get_grey

(**const unsigned char** *image,**int** width,**int** x,**int** y);

15: **float ***wb_camera_ get_range_image

(**WbDeviceTag** camera);

16: **float** wb_camera_range_image_ get_value(**const**

Float *rang image,**float** camer_near,**float** camera_far,**int**

Width , **int**x,**int**Y);

17: **void** wb_camera_move_window(**WbDeviceTag**

camer,**int**x,**int**Y);

18: **int** wb_camera_move_window(**WbDeviceTag**

camer,**const char** *filename,**int**quality);

توصیف دستورات

دستور شماره یک دوربین را جهت گرفتن تصاویر در هر میلی ثانیه ، فعال می کند. درستور شماره دو آن را غیر فعال می کند.

دستور شماره ۳، میدان دید field of view را به برنامه بر می گرداند و دستور ۴ آن را به مقداری که ما به برنامه می دهیم تنظیم می کند. مقدار ابتدایی زاویه دید در فیلد آن مشخص می شود.

با این حال دستور ۴ ، مقدار درون فیلد را تغییر نمی دهد. این دستور فقط بر جنبه کنترلی شبیه سازی تاثیر دارد.

دستورات ۵و ۶ پهنا و ارتفاع تصویر را به برنامه بر می گرداند.

دستور ۷ و ۸ مقدار دو فیلد نزدیک و دور گره دوربین را به برنامه بر می گرداند و در یک متغیر عددی قرار می دهد.

دستور شماره ۹ نوع دوربین را از پنجره درختی گرفته و به برنامه می دهد. مقدار بر گردانده شده از نوع عدد صحیح است و به صورت جدول ۵-۴ است.

جدول ۵-۴: مقدار دریافت شده از دستور شماره ۹

نوع دوربین **Type**	مقدار بر گردانده شده به برنامه
Color	WB CAMERA COLOR
Black and whit	WB CAMERA BLACK AND WHITE
Range-finder	WB CAMERA RANGE FINDER

دستور شماره ۱۰ آخرین تصویر گرفته شده بوسیله دوربین را دریافت می کند. این تصویر بصورت ترکیبی از پیکسل های قرمز ، سبز و آبی است. پیکسل ها در خطوط افقی ذخیره می شوند که از گوشه سمت چپ بالا به سمت گوشه سمت راست پایین مرتب می شوند. اندازه حافظه مورد استفاده برای این تصویر بر حسب بایت چنین بدست می آید:

Byte size = camera width × camera height × 3

دستور شماره ۱۱ تا ۱۳ می توانند مستقیما به سطح رنگهای قرمز ، سبز ، آبی (RGB) پیکسلها دسترسی داشته باشند. دستور شماره ۱۴ یعنی wb_camera_image_get_grey() مقدار سطح رنگ خاکستری پیکسل ها را بوسیله میانگین سه مولفه RGB به برنامه بر می گرداند. در کد نویسی به زبان c این چهار دستور (۱۱ تا ۱۴) از نوع unsigned char است و در بازه [0 , 255] قرار دارد. مثال c

…

```
Const unsigned char *image=

Wb_camera_get_image(camera);

For(int x =0; x<image_width;x++)

    For(int x =0; x<image_height;y++)  {
```

```
        Int r = wb_camera_image_get_red(image,
Image_width, x, y);
        Int g = wb_camera_image_get_green(image,
Image_width, x, y);
        Int b = wb_camera_image_get_blue(image,
Image_width, x, y);
        Printf ("red=%d, green=%d, blue=%d" ,r,g,b);
    }
```

.....

مثال شبیه سازی رباتی با یک دوربین در پوشه devices زیر مجموعه پوشه های وبتز قرار دارد.

دستور شماره ۱۵ ، هنگامی که دوربین فاصله یاب است استفاده می شود ، و محتویات آخرین تصویر گرفته شده بوسیله دوربین فاصله یاب را می خواند. تصویر برد دوربین بستگی به عمق تولید شده بوسیله کتابخانه گرافیکی دارد .

برای هر پیکسل ، فاصله شی تا دوربین را مشخص می کند. دستور ۱۶ را باید بعد از دستور شماره ۱۵ استفاده کرد تا فاصله با مانع بر حسب متر داده شود. تصویر برد دوربین به صورت یک ارایه رمز گزاری می شود که هر پیکسل عمق و فاصله آن پیکسل را تا دوربین نشان می دهد.

پیکسل به صورت خطی از سمت چپ پایین تا بالا سمت راست ذخیره می شوند. حافظه مورد استفاده این دستور نیاز به آزاد شدن ندارد چون بوسیله خود دوربین مدیریت می شود. اندازه این حافظه بر حسب بایت چنین بدست می آید.

Sizeof(float) × camera height × size= camera width

دستور شماره ۱۷ ، این دستور اجازه حرکت دادن پنجره تصاویر را در پنجره اصلی بوسیله مختصات y, x فراهم می کند.

دستور شماره ۱۷ ، این دستور اجازه حرکت دادن پنجره تصاویر را در پنجره اصلی بوسیله مختصات y,x فراهم می کند.

دستور ۱۸ گره دوربین ، اجازه ذخیره کردن تصاویری را که قبلاً توسط دستور wb_camer_get_image() گرفته شده است را می دهد. این دستور تصاویر را فقط در فرمت های JPEG , PNG می تواند ذخیره کند.

Filename نام ذخیره شده این فایل خواهد بود اگر پسوند این نام Png باشد فایلد یا این فرمت ذخیره می شود و اگر با پسوند jpg یا jpeg تمام شود با فرمت jpeg ذخیره می شود. پارامتر quality فقط برای فرمت jpeg مناسب است این پارامتر بین مقدار ۱ (بدترین کیفیت) تا ۱۰۰ (بهترین کیفیت) دامنه دارد. برای فرمت PNG پارامتر کیفیت نادیده فرض می شود.

مقدار بر گردان شده توسط این دستور 0 است اگر ذخیره تصویر با موفقیت صورت پذیرد و ۱- است اگر موفق به ذخیره نشود و مشکلی پیش آید.

۴-۱۲- گره فرستنده Emitter

گره فرستنده ، این گره برای مدل کردن فرستنده رادیویی ، سریال و مادون قرمز است . این گره می تواند به فیلد children ربات یا گره ناظر اضافه شود. گره فرستنده فقط برای فرستادن داده هاست و هیچ چیز را نمی تواند دریافت کند.

به منظور دست یافتن به یک ارتباط دو طرفه ، ربات نیاز به گره های فرستنده و دریافت کننده دارد.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	دامنه
SFString	Type	"radio"	"serial" یا "infra-red" یا "radio"
SFFloat	Range	-۱	-۱ یا مقادیر مثبت
SFFloat	maxRange	-۱	-۱ مقادیر مثبت
SFFloat	Aperture	-۱	-۱یا بین 0یا۲π
SFInt32	Channel	۰	
SFInt32	baudRate	-۱	-۱یا مقادیر مثبت
SFInt32	byteSize	۸	۸ یا بیشتر
SFInt32	bufferSize	۴۰۹۶	مقادیر مثبت

تعریف فیلدها

: type

نوع فرستنده را مشخص می کند که یکی از انواع radio (مقدار پیش فرض) serial,intra-red است. سیگنالهای رادیویی و سریال به مانع برخورد نمی کنند و از آنها می گذرند ، اما سیگنال مادون قرمز به موانع بر خورد می کند و ارتباط در صورت وجود مانع قطع می شود ، در حقیقت ارتباط مادون قرمزی باید چهره به چهره باشد. توجه کنید این فرستنده مادون قرمزی با سنسور فاصله مادون قرمز تفاوت دارد.

: range

شعاع کره امواج صادر شده است (بر حسب متر) ، یعنی برد فرستنده محسوب می شود ، یک گیرنده فقط وقتی می تواند امواج ارسالی را دریافت کند که در محدوده این کره قرار داشته باشد. مقدار ۱-(پیش فرض) به معنی برد و شعاع بی نهایت محسوب می شود.

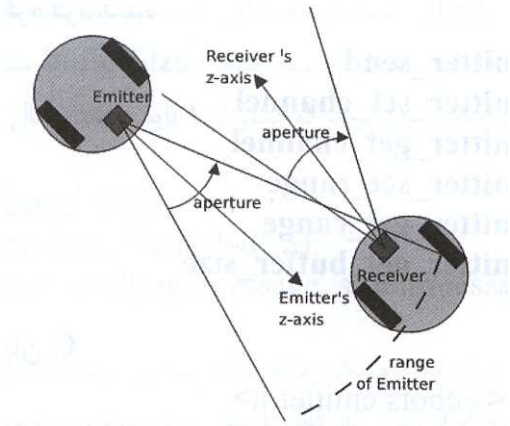
: max Range

مقدار ماکزیمم برد را مشخص می کند. این مقدار ماکزیمم مقداری را که در دستور emitter_set_rang() استفاده می شود را مشخص می کند. مقدار ۱- به معنی ماکزیمم برد است .

: aperture

این فیلد فقط برای فرستنده مادون قرمز استفاده می شود ، و زاویه باز شدگی هرم امواج ارسالی را مشخص می کند (به رادیان) . نوک هرم در مبدا دستگاه مختصات فرستنده (۰،۰،۰) قرار می گیرد. محور هرم نیز در راستای محور z دستگاه مختصات گره فرستنده قرار می گیرد ، یک فرستنده مادون قرمز فقط می تواند داده ها را به فرستنده ای ارسال کند که در این هرم باشد.

اگر این فیلد ۱- باشد (پیش فرض) زاویه در همه جهات هست و اصطلاحا omnidirectional است. شکل ۵-۸ را ببینید. برای دو نوع دیگر فرستنده ، این فیلد نادیده گرفته می شود.



شکل ۵-۸

: channel

طبیعتاً ، یک فرستنده و گیرنده برای تبادل داده ها باید از یک کانال یکسان استفاده کنند. با این حال کانال ۱- اجازه انتشار پیام به همه کانالها را می دهد.

کانال صفر (پیش فرض) به منظور ارتباط با ابزار جانبی فیزیک محیط در نظر گرفته شده است. برای ارتباط بین رباتها از مقادیر مثبت استفاده کنید.

: baudRate

این فیلد سرعت ارتباط را بر حسب بیت بر ثانیه مشخص می کند. مقدار ۱- یعنی سرعت بی نهایت و سبب رسیدن فوری اطلاعات از فرستنده به گیرنده می شود.

byteSize: اندازه بایت تعداد بیت های ارسالی در یک بایت اطلاعات را مشخص می کند.

:bufferSize

اندازه (بر حسب بایت) بافر انتقالی (حایل) را مشخص می کند. کل تعداد بایتها در بسته های صف شده فرستنده از عدد اندازه بافر نمی توانند بیشتر باشند.

دستورات گره فرستنده

1. wb_emitter_send

2. wb_emitter_set_channel

3. wb_emitter_get_ channel

4. wb_emitter_set_range

5- wb_emitter_get_range

6. wb_emitter_get_buffer_size

کد نویسی به زبان C

#include<webots/emitter.h>

1: **int** wb_emitter_send(**WbDeviceTag** emitter, **const**

Void ***data, **int size);

2: **Void** wb_emitter_set_channel(**WbDeviceTag**

emitter,**int** channel);

3: **int** wb_emitter_set_channel (**WbDeviceTag** emitter);

4: **Void** wb_emitter_set_range(**WbDeviceTag**

emitter,**double** range);

5: **double** wb_emitter_get_range(**WbDeviceTag**

emitter);

6: **int** wb_emitter_get_buffer_size(**WbDeviceTag**

emitter);

توصیف دستورات

دستور شماره یک ، به صف فرستنده یک بسته ای به اندازه پارامتر size اضافه می کند که این بسته در آدرسی که بوسیله پارامتر data که در دستور مشخص می شود ، قرار می گیرد.

بسته های اطلاعاتی صف بسته شده سپس با سرعتی که در فیلد baudRate مشخص شده به سمت گره گیرنده فرستاده می شوند (از صف فرستنده جدا می شوند).

اگر اطلاعات با موفقیت فرستاده شد عدد یک به برنامه بر گردانده می شود. اگر اطلاعات از صف فرستادن جدا نشدند (صف پر باشد) مقدار صفر بر گردانده می شود.

صف هنگامی پر شده در نظر گرفته می شود که مجموع بایتهای همه بسته های صف بندی شده بیشتر از مقدار بافر (حائل) بشود. که این مقدار بوسیله فیلد bufferSize مشخص می شود.

توجه کنید که یک بسته نمی تواند کوچکتر از ۱ باید باشد. مثال زیر را ببینید:

…

```
Char message [128];

Sprint(message, "hello%d",i);

Wb_emitter_send(tag, message, strlen(message)+1);
```

….

در مثال بالا ، بسته به فرم یک رشته تمام نشدنی فرستاده می شود. فرستنده و گیرنده روی نوع اطلاعاتی که قابل ارسال باشند هیچ محدودیتی قرار نمی دهند. هر کاربری شکل مناسب داده های خود را که با گیرنده و فرستنده مطابقت دارد انتخاب می کند.

دستور شماره ۲ ، به برنامه کنترلی اجازه تغییر کانال ارسال اطلاعات را می دهد. این کار فیلد channel زیر مجموعه گره فرستنده را تغییر می دهد.

بطور طبیعی یک فرستنده فقط اطلاعات را می تواند برای یک گیرنده ای بفرستد که کانال مشابه اش تنظیم شده باشد . با این حال دستور ویژه

WB_CHANNEL_BROADCAST اطلاعات را می تواند برای همه کانالها بفرستد. در یک فرستنده می توان با عوض کردن شماره کانال ، داده های انتخابی را به گیرنده های مختلف فرستاد. دستور شماره ۳ شماره کانال فرستنده را به برنامه بر می گرداند.

دستور شماره۴ فرستنده ، به برنامه کنترلی اجازه می دهد برد ارسال داده ها را هنگام اجرا تغییر دهد. بسته داده ها فقط به گیرنده ای می رسند که در برد فرستنده قرار گرفته باشد.

این دستور فیلد وابسته به برد یعنی RANGE را نیز تغییر می دهد. اگر مقدار داده شده به برد در این دستور بیشتر از فیلد ماکزیمم برد بود ، برنامه برد را به اندازه فیلد maxRange تغییر می دهد.

دستور شماره پنج نیز برد فعلی گیرنده را به برنامه بر می گرداند. در هر دو دستور ۴ و ۵ مقدار۱- برابر برد بی نهایت است.

دستور شماره ۶: اندازه بافر را بر حسب بایت به برنامه بر می گرداند. این مقدار همان مقدار فیلد bufferSize گره فرستنده است. فیلد اندازه بافر ، ماکزیمم تعداد بایتها را که صف فرستنده می تواند نگه دارد ، مشخص می کند.

هنگامی که بافر پر شود ، فراخواندن دستور wb_emitter_send_packet() با عدم موفقیت مواجه می شود و مقدار صفر را می دهد.

۴-۱۳- گره گیرنده Receiver

گره گیرنده برای مدل کردن گیرنده های رادیویی ، مادون قرمز و سریال استفاده می شود. این گره باید به فیلد children یک ربات یا گره ناظر اضافه شود. یک گیرنده فقط می تواند اطلاعات را دریافت کند و چیزی نمی تواند بفرستد . به منظور دستیابی به ارتباط دو طرفه ، ربات نیاز به هم فرستنده و هم گیرنده دارد.

جدول فیلدها

نوع متغیر	نام فیلد	مقدار پیش فرض	دامنه
SFString	Type	"radio"	"serial" یا "infra-red" یا "radio"
SFFloat	Aperture	-۱	۱- یا بین ۰تا۲π
SFInt32	Channel	۰	۱-یا بین 0تا ∞
SFInt32	baudRate	-۱	۱- یا مقادیر مثبت
SFInt32	byteSize	۸	۸ یا بیشتر
SFInt32	bufferSize	۴۰۹۶	مقادیر مثبت

تعریف فیلدها

- **type :**

نوع گیرنده را مشخص می کند که می تواند Serial , radio یا infra-red باشد. نوع رادیویی پیش فرض است . همچنین در نوعهای سریال و رادیویی اجسام ، مانع ارتباط بین دو ربات نمی توانند بشوند ولی نوع گیرنده مادون قرمز به هر نوع جسمی یا رباتی بر خورد می کند.

- **aperture :**

این فیلد فقط در نوع مادون قرمز زاویه باز شدگی هرم امواج دریافتی را بر حسب رادیان مشخص می کند. گیرنده فقط می تواند از فرستنده هایی که در هرم دریافت آن قرار دارد ، پیام بگیرد. نوک این هرم در نقطه [0 0 0] دستگاه مختصات متصل به گیرنده قرار دارد و محور هرم در راستای محور z دستگاه مختصات گیرنده است.

اگر این فیلد مقدار ۱- (پیش فرض) را داشته باشد به این معنی است که سیگنال از تمام جهات دریافت می شود و محدودیتی ندارد. این فیلد برای نوع های سریال و رادیویی نادیده فرض می شود.

- **channel :**

کانال دریافت را مشخص می کند. این عدد برای گیرنده مادون قرمز یک عدد مشخصه و برای گیرنده رادیویی یک فرکانس را مشخص می کند. هر دوی گیرنده و فرستنده باید در یک کانال تنظیم باشند. با این حال مقدار ۱- به گیرنده اجازه می دهد در وضعیت دریافت همه کانالها قرار گیرد.

- **baudRate :**

سرعت ارتباط بر حسب بیت بر ثانیه است که با مقدار فیلد مشابه در فرستنده باید یکی باشد.

- **byteSize :**

تعداد بیت های تشکیل دهنده یک بایت را مشخص می کند (معمولا ۸ است ولی ممکن است برای کنترل بیت استفاده شود). این فیلد باید با فیلد مشابه در فرستنده یکسان باشد.

- **bufferSize :**

اندازه بافر پذیرنده بر حسب بایت است . اندازه داده های دریافت شده نباشد در هیچ زمان از اندازه بافر بیشتر باشد ، در غیر اینصورت داده ها از دست می روند. اگر هنگامی که داده های جدید می رسند ، داده های گذشته خوانده نشده باشند، داده های قبلی از بین می روند.

دستورات گره گیرنده Receiver

1. wb_receiver_enable

2. wb_receiver_disable

3. wb_receiver_get_queue_length

4. wb_receiver_next-packet

5. wb_receiver_get_data

6. wb_receiver_get_data-size

7. wb_receiver_get_signal_strength

8. wb_receiver_get emitter_direction

9. wb_receiver_set_channel

10. wb_receiver_get_channel

کد نویسی به زبان C

1: #include<webots/receiver.h>

1:**void** wb_receiver_enable(**WbDeviceTag** receiver, **int**

Ms);

2: **void** wb_receiver_disable(**WbDeviceTag** receiver);

3:**int** wb_receiver_get_queue_length(**WbDeviceTag**

receiver);

4: **void** wb_receiver_next_packet(**WbDeviceTag**

receiver);

5: **const void** ***wb_receiver_next_packet(**WbDeviceTag**

receiver);

6: **int** wb_receiver_get_data_size(**WbDeviceTag**

receiver);

7: **double** wb_receiver_get_signal_strength

(**WbDeviceTag** receiver);

8: **const double** ***wb_receiver_get_emitter_direction

(**WbDeviceTag** receiver);

9: **void** wb_receiver_set_channel(**WbDeviceTag**

Receiver,int channel);

10: **int** wb_receiver_get_channel(**WbDeviceTag**

receiver);

توصیف دستورات

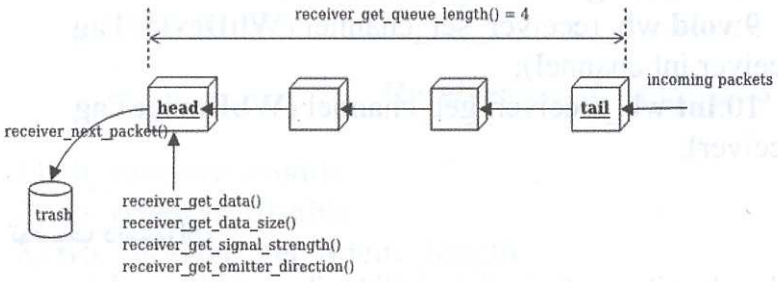
دستور ۱ گیرنده را برای دریافت اطلاعات فعال می کند. دریافت داده ها در پس زمینه حلقه برنامه کنترلی هر ms ثانیه صورت می گیرد. داده های که تازه رسیدند در انتهای صف پذیرش قرار می گیرند (شکل ۵-۹) . داده های تازه رسیده اگر از اندازه بافر مقدارشان بیشتر باشد نادیده گرفته می شوند.

به منظورپرهیز از پر شدن بافر باید داده ها با سرعتی به اندازه کافی بالا خوانده شوند. دستور شماره ۲ گیرنده را غیر فعال می کند.

دستور شماره ۳ تعداد بسته های اطلاعاتی را که اکنون در صف دریافت هستند را نمایش می دهد. دستور ۴ بسته جلویی را پاک می کند و بسته بعد از آن سر یا جلو صف قرار می گیرد. هنگامی که صف خالی باشد دستور ۴ اگر اجرا شود نادیده گرفته می شود. به مثال زیر توجه کنید:

…

```
While (wb_receiver_get_queue_length(tag)>0){
Const char *message= wb_receiver_get_data(tag);
Const float *dir=
wb_receiver_get_emitter_direction(tag)
double signal= wb_receiver_get_signal_strength(tag);
printf("received:%s (signal=%g, dir=[%g %g %g]\n",
message, signal, dir[0], dir[1], dir[2]);
wb_receiver_next_packet(tag);
```



شکل ۵-۹

در این مثال فرض شده است که پیام به صورت یک رشته تمام نشدنی فرض شده است. فرستنده و گیرنده هیچ محدودیتی بر نوع اطلاعاتی که می توان فرساتد ایجاد نمی کنند.

فرستنده و گیرنده در وبتز بسته ها را به همان ترتیبی که ایجاد شده است با اطمینان انتقال می دهند ولی با این حال زمان بندی ویژه ای برای انتقال آنها ندارد. بعضی وقتها چندین بسته ممکن است جمع بشوند و با هم برسند. مثلاً ، در نظر بگیرید که دو ربات دارای فرستنده و گیرنده باشند.

اگر هر دو ربات از یک گام زمانی یکسان استفاده کنند و هر کدام یک بسته در هر گام دریافت می کنند ، اما گاهی ممکن است هیچ بسته ای نیز دریافت نکنند ، و گاهی دو بسته دریافت کنند. دستور ۳ برای دانستن تعداد واقعی صف گیرنده مفید است.

دستور شماره ۵ گیرنده ، بسته اطلاعاتی جلو صف پذیرش را به برنامه بر می گرداند. مقدار بر گردانده شده تا زمانی معتبر است که داده بعدی با دستور wb_receiver_next_packet() فرا خوانده شود. هنگامی که صف خالی است غیر منطقی است که دستور ۵ اجرا شود.

دستور ۶ تعداد بایتهای موجود در بسته جلویی صف را که بر برنامه می دهد. این اندازه هموار با اندازه داده در فرستنده برابر است.

دستور ۷ گیرنده، روی بسته جلویی صف دریافت عمل می کند (شکل ۵-۹) . این دستور قدرت یک سیگنال شبیه سازی شده را به برنامه بر می گرداند. قدرت سیگنال برابر با معکوس مربع فاصله بین فرستنده و گیرنده . به عبارت دیگر $s=1/r^2$ که r فاصله میان گیرنده و فرستنده است.

دستور ۸ نیز روی بسته جلویی صف عمل می کند. این دستور یک برنامه یکه (نرمال شده) که جهت فرستنده را نسبت به دستگاه مختصات گیرنده مشخص می کند را بر می گرداند. سه مولفه این بردار

اشاره به جهتهای x, y, z دارد. در بسیاری از کاربردهای ۲ بعدی حرکت ربات در صفحه x,z است بنابراین جهت مثبت x اشاره دارد که فرستنده سمت چپ گیرنده قرار دارد.

دستور ۹ کانال گیرنده را تغییر می دهد. توجه کنید که در صورت تغییر کانال ، گیرنده فقط اطلاعات کانالی که روی آن تنظیم شده است را دریافت می کند. دستور ۱۰ نیز شماره کانال فعلی گیرنده را به برنامه کنترلی بر می گرداند.

۴-۱۴- گره ناظر Superviser

سوپروایزور (ناظر) یک برنامه ای است که جهان مجازی و رباتها را کنترل می کند. برای راحتی بصورت یک ربات بدون چرخ ارائه می شود که بوسیله برنامه کنترل می شود و توانایی های دارد که قادر به کنترل کل جهان مجازی است .

مثلا جا به جا کردن اشیا یا رباتی خاص ، عوض کردن رنگ و سایر خصوصیات جهان ، تغییر زاویه دید ، نمایش زمان گرفتن عکس و غیره .

برنامه کنترلی ناظر یک نوع خاصی از برنامه کنترلی ربات است . بنابراین دستورات wb_robot_init() , wb_robot_step() نیز در برنامه کنترلی ناظر نیز باید اعمال شود. چونکه ناظر می تواند شامل سنسور و محرک ها در فیلد children خود باشد بنابراین دستورات مرتبط با آنها نیز در برنامه کنترلی آن مورد استفاده قرار می گیرد.

دستورات گره ناظر (بخش اول)

1. wb_suprvisor_export_image

2. wb_suprvisor_node_get_from_def

3. wb_suprvisor_node_get_root

4. wb_suprvisor_node_get_rype

5. wb_suprvisor_node_get_name

6. wb_suprvisor_node_get_field

7. wb_suprvisor_node_set_label

8. wb_suprvisor_simulation_quit

9. wb_suprvisor_simulation_revert

10. wb_suprvisor_simulation_physics_reset

کد نویسی به زبان C

```
#include<webots/supervisor.h>

1: void wb_suprvisor_export_image(const char

*filename, int quality);

2: WbNodeRef wb_suprvisor_node_get_form_def

(const char*def);

3 : WbNodeRef wb_suprvisor_node_get_root();

4: WbNodeRef wb_suprvisor_node_get-type

(WbNodeRef node);

5: const char *wb_suprvisor_node_get_name

(WbNodeRef node);

6: WbFieldRef wb_suprvisor_node_get_field

(WbNodeRef node,const

Char *field_name);

7: void wb_suprvisor_set_label(int id,const char

*text,double x ,duble

y ,duble size,int  color,duble transparency);
```

8: **void** wb_suprvisor_simulation_quit ();

9: **void** wb_suprvisor_simulation_revers ();

10: **void** wb_suprvisor_simulation_physics_reset ();

توصیف دستورات گره ناظر

دستور شماره یک ، تصویر جاری پنجره اصلی وبتز را با نام filename و یا فرمت JPEG ذخیره می کند. پارامتر quality (در محدوده ۱ تا ۱۰۰) کیفیت عکس را مشخص می کند. پارامتر filename باید به یک نام معتبر اشاره کند.

(مطلق یا نسبی) مثلا images/snapshot.jpg – sanapshot.jpg . در حقیقت ابتدا یک فایل موقعت ذخیره می شود سپس نام آن به آن نامی که در دستور آمده تغییر می کند. مثالی درباره ذخیره کردن عکس در آدرس

projects/samples/howto/worlds/supervisor.wbt وجود دارد. در این مثال گره ناظر هر زمانی که گلی به ثمر می رسد ، عکسی از پنجره شبیه سازی می گیرد و ذخیره می کند.

دستور شماره ۲ ، یک گره را از جهان مجازی بوسیله نام DEF آن به برنامه کنترل بر می گرداند. مقدار بر گردانده شده در متغیری از نوع WbNodeRef قرار می گیرد که جایگزین نام گره می شود هر جا که قرار باشد فرا خوانی شود. اگر گره خوانده شد وجود نداشته باشد ، دستور مقدار خالی را بر می گرداند.

دستور ۳، عنوان گره های ریشه ای که در واقع یک گره گروهی هستند و در پنجره درختی در سطح بالایی قرار دارند را به برنامه بر می گرداند. گره ریشه ای یک فیلد چند گره ای (MFN) به نام children دارد که برای خواندن بقیه گره های از آن استفاده می شود. گره ریشه ای گره اصلی برنامه مثل ربات یا ناظر است. مثالی با عنوان supervisor.wbt در این رابطه در آدرس زیر قرار دارد:

Projects/samples/devices/worlds

دستور ۴، یک مقدار نمادین در ارتباط با نوع گره بر می گرداند. لیست انواع گره ها در webots/nodes.h موجود است. انواع گره مانند چراغ ، سرو و .. است . مقدار بر گردانده شده مانند:

WB_NODE_DIFFERENTIAL_WHEELS است.

دستور ۵ یک مقدار رشته ای شامل نام گره را بر می گرداند و در یک متغیر قرار می دهد.

دستور شماره ۶ ، یک فیلد از یک گره را بر می گرداند. این فیلد با نامش filed_name مشخص می شود. این فیلد می تواند یک فیلد تنها (SF) با یک فیلد چندتایی باشد (MF) . اگر نامی که برای فیلد ذکر می شود وجود نداشته باشند، مقدار خالی بر گردانده می شود.

دستور۷ ، یک نوشته یا بر چسب را در پنجره اصلی نمایش نشان می دهد. اولین پارامتر این دستور id یا شناسه بر چسب است. این شناسه عددی است بین ۰ تا ۶۵۵۳۵ که می توان با این شناسه ، بر چسب را فراخوانی یا عوض کرد.

پارامتر دوم متنی که باید نمایش داده شود است. تعداد حروف آن تا ۱۲۷ است. پارامتر سوم و چهارم مختصات متنی است که نسبت به گوشه بالا و سمت چپ پنجره اصلی اندازه گیری می شود و به صورت درصد است ، یعنی مقادیر x , y بین صفر تا یک است.

پارامتر پنجم ، اندازه متن را مشخص می کند، پارامتر ششم رنگ متن است که بصورت متغیر ۴ باتی RGB نشان داده می شود و بایت اول نادیده گرفته می شود ، باید دوم رنگ قرمز ، بایت سوم رنگ سبز و بایت چهارم رنگ آبی است.

پارامتر آخر شفافیت متن بر چسب شده است. عدد صفر عدم شفافیت و عدد یک به معنی شفافیت کامل است (متن نامرئی می شود).

مثال:

```
Wb_supervisor_set_label(0,"hello

World",0,0,0.1,0xff0000,0);
```

این دستور متن heloo world را به رنگ آبی در گوشه سمت چپ و بالا پنجره نشان می دهد.

دستور شماره ۸ ، این دستور یک پیام به قسمت شبیه سازی وبتز می فرست و فرایند شبیه سازی را فوری متوقف می کند. پس از این دستور تمام فرایند کنترلی نیز متوقف می شود.

دستور ۹ ، به موتور شبیه سازی پیامی می فرستد و فرایند شبیه سازی را دوباره از اول اجرا می کند.

دستور شماره ۱۰، با فرستادن پیامی به موتور شبیه سازی حرکت تمام اجسام فیزیکی را متوقف می کند. یعنی تمام گره solid ها را که شامل physics باشند را سرعت خطی و زاویه آنها را صفر می کند بنابراین اینرسی نیز صفر می شود.

این عمل با فراخوانی دستور موتور باز دینامیکی (ODE) ، ()dBodySetLinearVel و ()dBodySetAngularVel برای اجسام نیز انجام می شود. این دستور برای بازرگردانی ربات به موقعیت و اینرسی اولیه نیز مفید است.

دستورات گره ناظر (بخش دوم)

11. wb_ supervisor _start_animation

12. wb_ supervisor _stop_animation

13. wb_ supervisor _start_movie

14. wb_ supervisor _stop_movie

15. wb_sup_wb_supervisor_field_get_type

16. wb_ supervisor_field_get_type_name

17. wb_ supervisor_field_get_countervisor_stop_movie

18. wb_ supervisor_field_get_sf_bool

19. wb_ supervisor_field_get_sf_int32

20. wb_ supervisor_field_get_sf_float

21. wb_ supervisor_field_get_sf_vec2f

22. wb_ supervisor_field_get_sf_vec3f

23. wb_ supervisor_field_get_sf_rotation

24. wb_ supervisor_field_get_sf_color

25. wb_ supervisor_field_get_sf_string

26. wb_ supervisor_field_get_sf_node

27. wb_ supervisor_field_get_mf_bool

28. wb_ supervisor_field_get_mf_int32

29. wb_ supervisor_field_get_mf_float

30. wb_ supervisor_field_get_mf_vec2f

31. wb_ supervisor_field_get_mf_vec3f

32. wb_ supervisor_field_get_mf_rotation

33. wb_ supervisor_field_get_mf_color

34. wb_ supervisor_field_get_mf_srting

35. wb_ supervisor_field_get_mf_node

36. wb_ supervisor_field_set_sf_bool

37. wb_ supervisor_field_set_sf_float

38. wb_ supervisor_field_set_sf_vec2f

39. wb_ supervisor_field_set_sf_vec3f

40. wb_ supervisor_field_set_sf_rotation

41. wb_ supervisor_field_set_sf_color

42. wb_ supervisor_field_set_sf_string

43. wb_ supervisor_field_set_mf_bool

44. wb_ supervisor_field_set_mf_int32

45. wb_ supervisor_field_set_mf_float

46. wb_ supervisor_field_set_mf_vec2f

47. wb_ supervisor_field_set_mf_vec3f

48. wb_ supervisor_field_set_mf_ratation

49. wb_ supervisor_field_set_mf_color

50. wb_ supervisor_field_set_mf_string

51. wb_ supervisor_field_set_mf_node

52. wb_ supervisor_field_import-mf_node

کد نویسی به زبان C

```
#include<webots/supervisor.h>

11: void wb_supervisor_start_animation (const char

*filename);

12: void wb_supervisor_stop_animation ();

13: void wb_supervisor_start_movie (const char

*filename,int width, int height, int type,int quality);

14: void wb_supervisor_stop_movie ();

15: void wb_supervisor_field-get_type

(WbFieldRef field);

16: vonst char  * wb_supervisor_field_get_type_name

(WbFieldRef field);

17: int wb_supervisor_field_get_count(WbFieldRef field);

18: bool wb_supervisor_field_get_sf_bool(WbFieldRef

field);
```

19: **int** wb_supervisor_field_get_sf_int32 (**WbFieldRef**

field);

20: **double** wb_supervisor_field_get_sf_float (**WbFieldRef**

field);

21: **const double** *wb_supervisor_field_get_sf_vec2f

(**WbFieldRef** sf_field);

22: **const double** *wb_supervisor_field_get_sf_vec3f

(**WbFieldRef** field);

23: **const double** *wb_supervisor_field_get_sf_rotation

(**WbFieldRef** field);

24: **const double** *wb_supervisor_field_get_sf_color

(**WbFieldRef** field);

25: **const char** *wb_supervisor_field_get_sf string

(**WbFieldRef** field);

26: **WbNodeRef** wb_supervisor_field_get_sf_node

(**WbFieldRef** field);

27: **bool***wb_supervisor_field_get_mf_bool(**WbFieldRef**

Field,**int** index);

28: **int***wb_supervisor_field_get_mf_in32(**WbFieldRef**

Field,**int** index);

29: **double** wb_supervisor_field_get_mf_float(**WbFieldRef**

Field,**int** index);

30: **const double** *wb_supervisor_field_get_mf_vec2f

(**WbFieldRef** Field,**int** index);

31: const double *wb_supervisor_field_get_mf_vec3f

(WbFieldRef Field,int index);

32: **const double** *wb_supervisor_field_get_mf_rotation

(**WbFieldRef** Field,**int** index);

33: **const double** *wb_supervisor_field_get_mf_color

(**WbFieldRef** Field,**int** index);

34: **const char** *wb_supervisor_field_get_mf_string

(**WbFieldRef** Field,**int** index);

35: **WbNodeRef***wb_supervisor_field_get_mf_vec2f

(**WbFieldRef** Field,**int** index);

36: **void** wb_supervisor_field_set_sf_bool(**WbFieldRef**

Field,**bool** value);

37: **void** wb_supervisor_field_set_sf_int32(**WbFieldRef**

Field,**int** value);

38: **void** wb_supervisor_field_set_sf_float(**WbFieldRef**

Field,**double** value);

39: **void** wb_supervisor_field_set_sf_vec2f(**WbFieldRef**

Sf_Field,**const double** values[2]);

40: **void** wb_supervisor_field_set_sf_vec3f(**WbFieldRef**

Field,**const double** values[3]);

41: **void** wb_supervisor_field_set_sf_vec2f(**WbFieldRef**

Sf_Field,**const double** values[2]);

42: **void** wb_supervisor_field_set_sf_color(**WbFieldRef**

Field,**const double** values[3]);

43: **void** wb_supervisor_field_set_sf_string(**WbFieldRef**

Field,**const double** *value);

44: **void** wb_supervisor_field_set_mf_bool(**WbFieldRef**

Field **,int** index,**bool** value);

45: **void** wb_supervisor_field_set_mf_int32(**WbFieldRef**

Field,**int** index,int value);

46: **void** wb_supervisor_field_set_mf_float(**WbFieldRef**

Field,**int** index,**double** value);

47: **void** wb_supervisor_field_set_mf_vec2f(**WbFieldRef**

Field,**int** index,**const double** value[2]);

48: **void** wb_supervisor_field_set_mf_vec3f(**WbFieldRef**

Field,**int** index,**const double** value[3]);

49: **void** wb_supervisor_field_set_mf_rotation(**WbFieldRef**

Field,**int** index,**const double** value[4]);

50: **void** wb_supervisor_field_set_mf_color(**WbFieldRef**

Field,**int** index,**const double** value[3]);

51: **void** wb_supervisor_field_set_mf_string(**WbFieldRef**

Field,**int** index,**const double** *value);

52: **void** wb_supervisor_field_import_mf_node

(**WbFieldRef** Field,**int** position,**const double** *filename);

توصیف دستورات گره ناظر (بخش دوم)

دستور ۱۱ ، شروع به ذخیره یک فایل انیمشن از صفحه اصلی شبیه سازی می کند و هنگامی فایل را بطور کامل در ادرسی ذخیره می کند که دستور توقف (دستور۱۲) اجرا شود. این فایل با نامی که در پارامتر filename نوشته می شود با فرمت wva ذخیره می شود. این فایل را می توان با نرم افزار Webviev مشاهده کرد این نرم افزار بطور مجانی از سایت cyberbotics.com قابل دانلود است.

دستور ۱۳ ، شروع به فیلمبرداری از محیط شبیه سازی می کند تا هنگامی که دستور ۱۴ اجرا شود. سپس فایل با پسوند avi و با نامی که در دستور مشخص می شود ذخیره می گردد.

دستور ۱۵، نوع فیلدی را که قبلا توسط دستور wb_suprvisor_node_get_field() گرفته شده است را به برنامه بر می گرداند. این نوعها در webots/supervisor.h تعریف شده اند. مثلا WB SF WB SF STRING,WB MF NODE, FLOAT

دستور ۱۶ ، یک مقدار رشته ای بر می گرداند که مشخص کننده نوع متغیر فیلد است مثلا SFFloat

دستور ۱۷ ، تعداد آیتم های یک فیلد چند گانه (MF) را بر می گرداند.

دستورات شماره ۱۸ تا ۲۶ مقادیر یک فیلد یگانه (SF) را بر می گردانند. مثلا رنگ و غیره

دستورات ۲۷تا۳۵ مانند مقادیر فیلدهای چندگانه را بر می گردانند و اندیس شماره فیلد را از بالا مشخص می کند. اندیس از صفر شروع تا تعداد ایتمها منهای یک . دستور شماره ۳۶ تا ۴۳ به یک فیلد یگانه مقدار می دهد.

دستور ۴۴ تا ۵۱ ، به ایتمهای یک فیلد چند گانه مقدار می دهد. در این دستور اندیس وجود دارد که مانند قبل از صفر تا کل فیلدها منهای یک است.

مثال texture change.wbt نحوه تغییر عکس توسط ناظر را نشان می دهد. مثال

Soccer.wbt نحوه فراخوان فیلدها و تنظیم آنها را نشان می دهد.

دستور شماره ۵۲ ، یک گره با فیلد چند گانه را به محیط وبتز وارد می کند.

این گره باید به صورت wbt. تعریف شود. این نوع گره بوسیله دکمه Export در پنجره درختی تولید می شود. پارامتر position جایی را که گره باید قرار گیرد ، مشخص می کند. این مقدار می تواند مثبت یا منفی باشد.

0: در ابتدای پنجره درختی قرار می گیرد.

1 : در موقعیت دوم قرار می گیرد.

2: به عنوان آخرین گره در لیست قرار می گیرد.

3-: از اخر دومین گره می شود.

به همین ترتیب سایر اعداد جایگاه گره وارد شده در لیست را مشخص می کنند.

پارامتر filename می تواند بصورت یک آدرس نسبی یا منطقی در نظر گرفته شود.

منابع

http://www.royak.ir/robotics-book/49-farsi-book/75----webots.html

قنواتی، مهدی ، راهنمای جامع Webots