Penn Engineering

GRASP Laboratory

Darwin-OP Software Example

Stephen McGill, University of Pennsylvania
Seung-Joon Yi, University of Pennsylvania

Penn Engineering

GRASP Laboratory

Robot Basics

Connecting to Darwin-OP, Running the Soccer programs, Understanding the code structure

Penn Engineering

GRASP Laboratory

Connecting to Darwin-OP

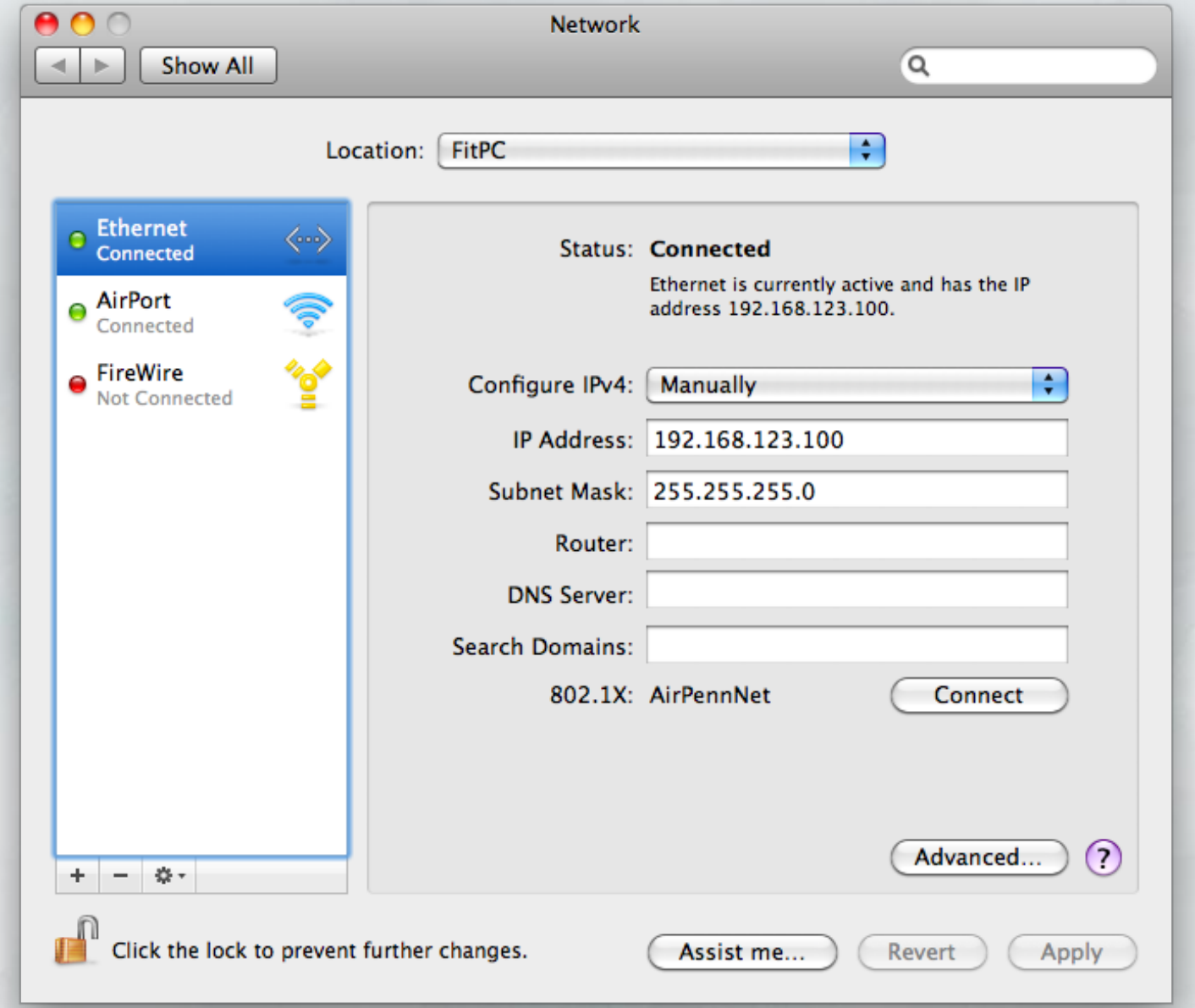
- Darwin runs on Ubuntu Linux (adaptable to other systems)
- Connection established using VNC or ssh

Darwin-OP's Current Configuration:
Wired ethernet IP address: 192.168.123.1

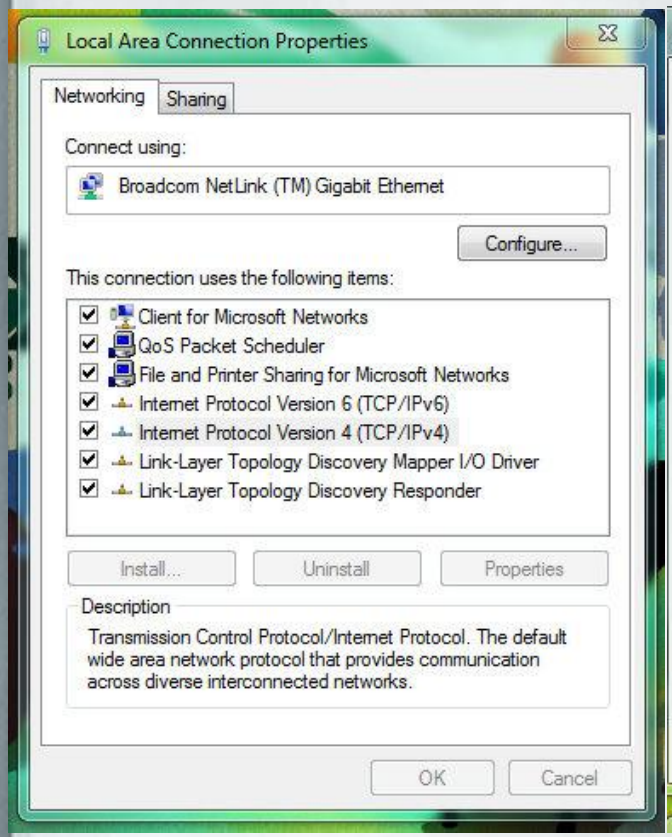
Let’s set up your computer to talk with Darwin, accordingly.

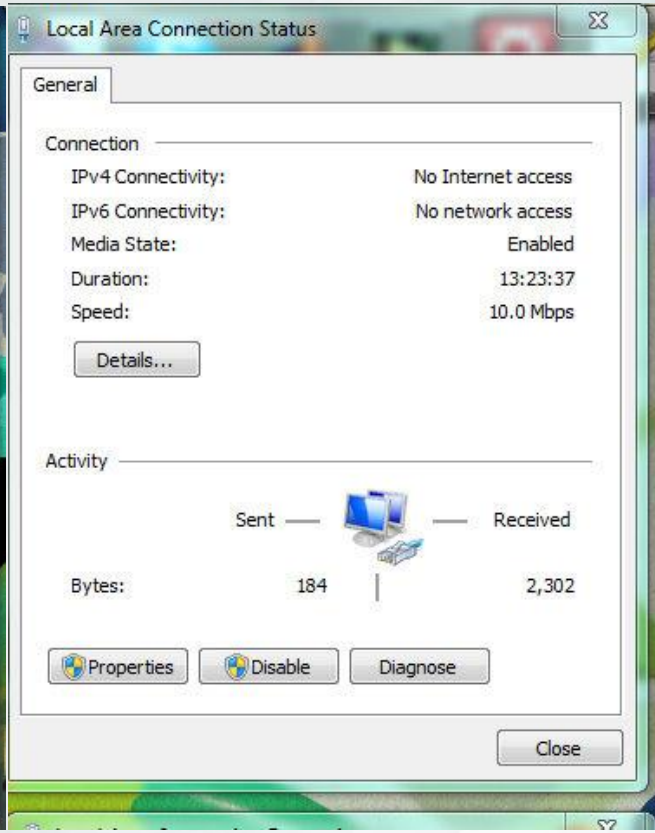
Set your IP Address to 192.168.123.100

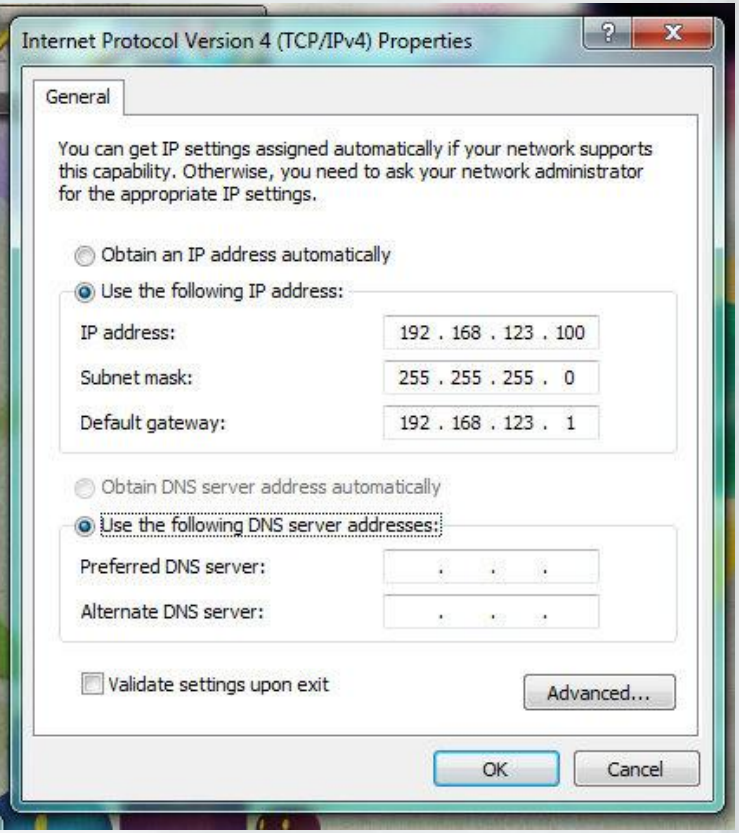
Network Connections



Network Connections







Penn Engineering

GRASP LABORATORY

Connecting to Darwin-OP

After turning the robot on, and waiting for a minute...

VNC Connection (Recommended, we will give you the installer)

1. Open your VNC client (tightvnc, Chicken of the VNC, etc.)
2. Connect to 192.168.123.1
3. You're in!

SSH Connection (If VNC not possible)

1. Open your favorite terminal or ssh client (putty)

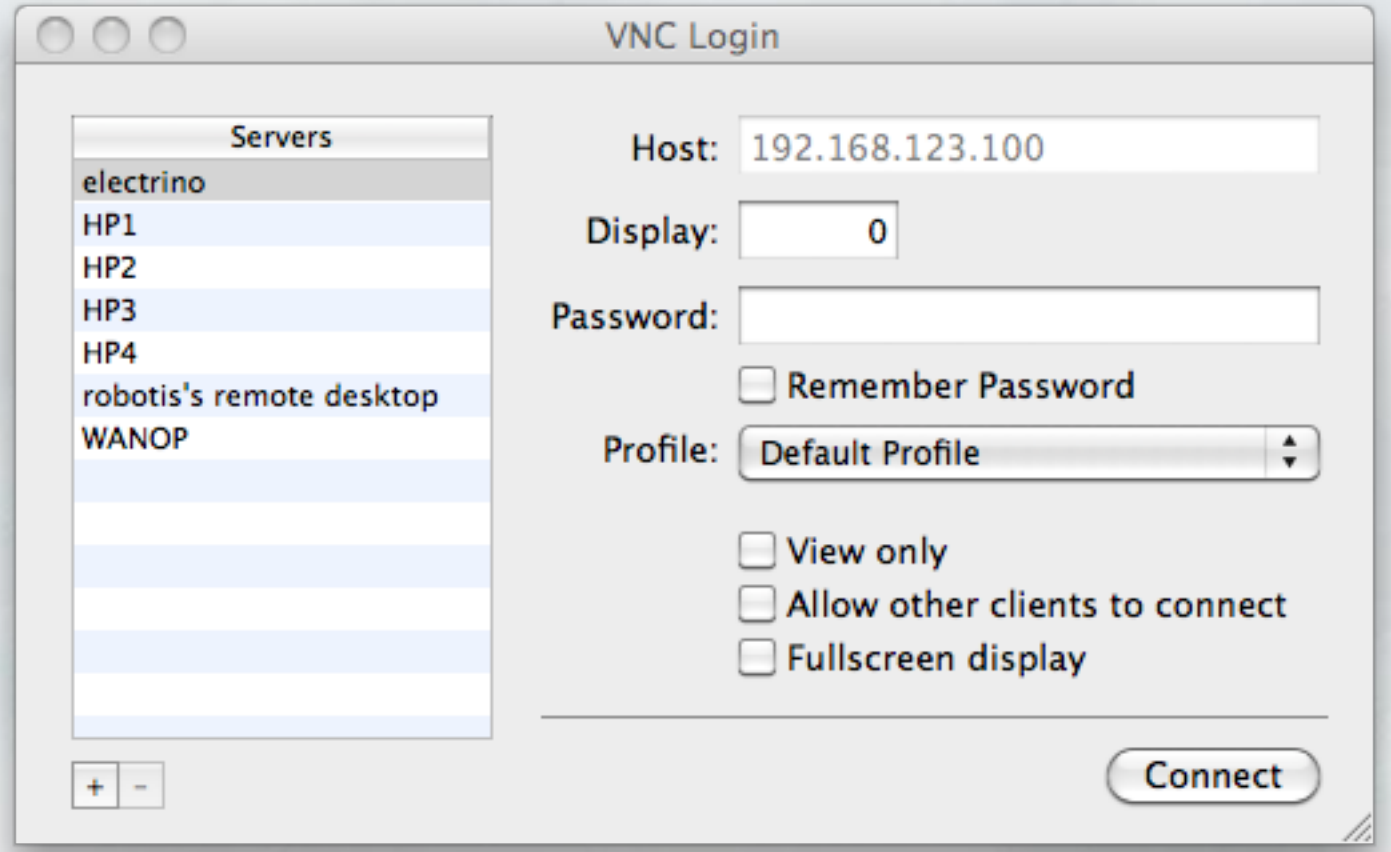
– Connect to 192.168.123.1
– You're in!

Username: robotis
Password: 111111 (six 1's)

Penn Engineering

GRASP LABORATORY

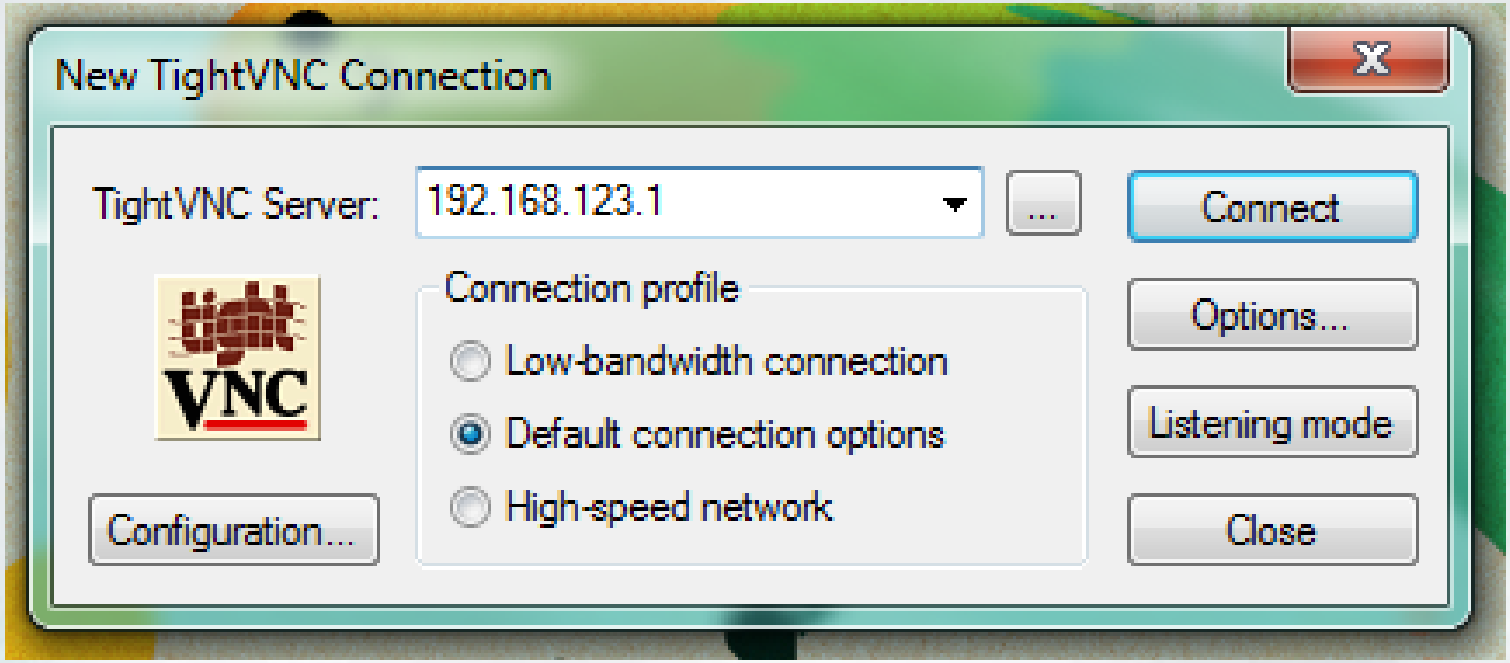
VNC Connection



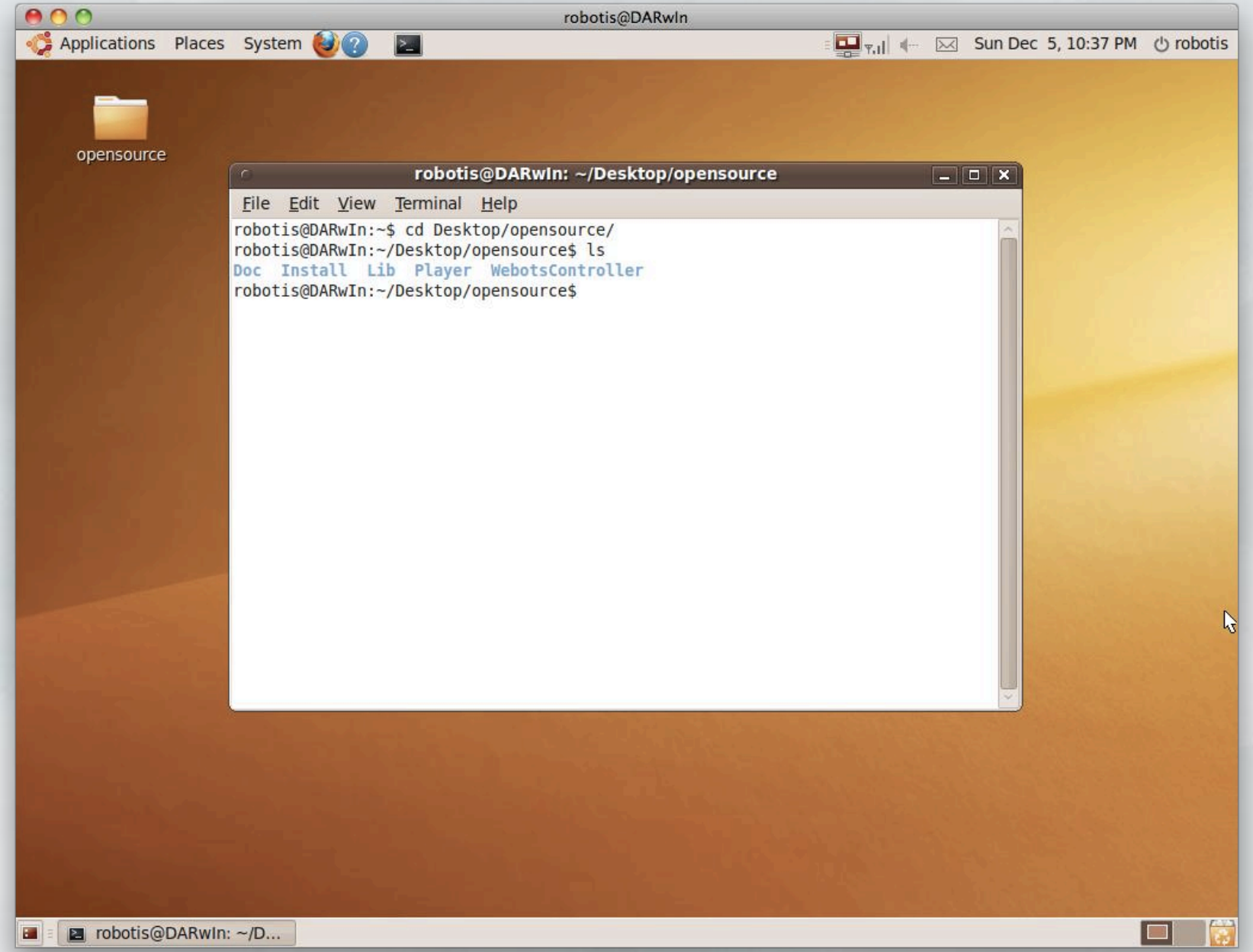
Penn Engineering

GRASP LABORATORY

VNC Connection



VNC Connection



Running the Demo program

We're all in this for the kicks, so let's get a taste of Darwin-OP movin' and groovin'

1. Open a terminal on Darwin-OP (ssh users ignore)
– Execute “cd Desktop/~~opensource~~/Player”
– Execute “screen -S dcm”
– Execute "cd Lib"
– Execute "lua run_dcm.lua"

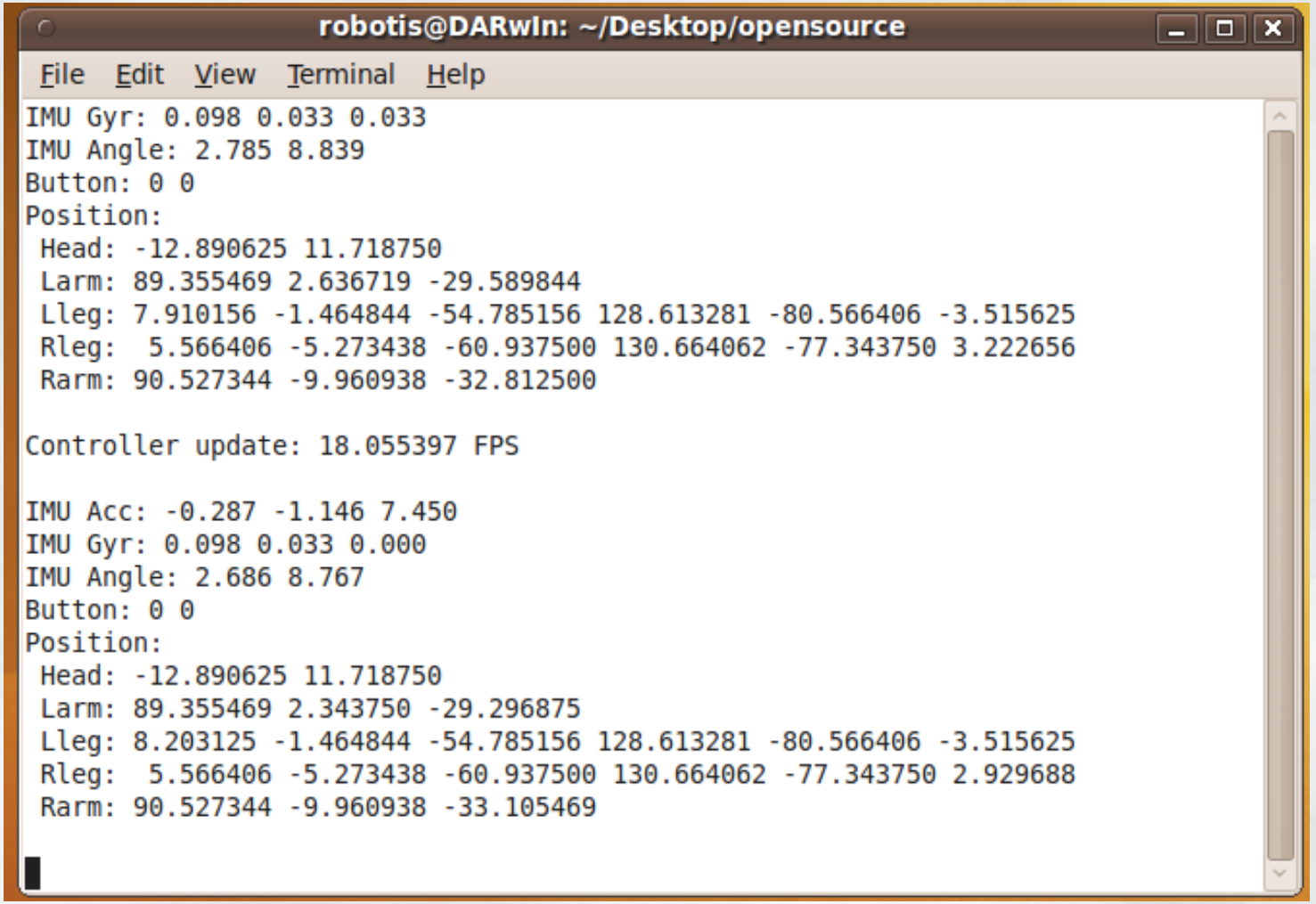
upenn_humanoid_1.1

This is the motor and sensor updating process. Move the Darwin-OP's joint around, and watch the numbers change!

Penn Engineering

GRASP LABORATORY

Demo Program



Running the Demo program

Move the head around, one axis at a time, and see how the DCM output changes.

Shake Darwin-OP's hand, and see the numbers change

Running the Demo program

Press "Ctrl-a Ctrl-d" to let this process continue, but as a background job. Now, let's have Darwin-OP move on its own.

FIRST: place Darwin-OP in a kneeling position

-Execute "lua demo.lua”

This is the "high level" demo process. Darwin-OP should stand up, march in place, wave, and sit down.

Code Structure

It's alive! Awesome - but let's see how Darwin-OP ticks.

- Still in the "opensource/Player" directory, execute "ls" to see the folders.

There's lot of stuff there, but the directories just organize the various module's of Darwin-OP's brain.

The Vision directory contains image processing code, the BodyFSM directory contains finite state machine code for where Darwin-OP wants to move, etc.

We will analyze these one by one!

Motion

Keyframing, Locomotion, Body state machine

Limb Motion

First, we're going to make Darwin-OP move its head around.

1. Check that your DCM process is still running:
   - Execute "screen -r dcm" If all is good, hit "Ctrl-a Ctrl-d"
2. Change into the "Lib" directory: "cd Lib"

– Execute "lua" and follow the yellow brick code on the next page...

Limb Motion

> b = require 'DarwinOPBody' --This provides I/O with DCM
> b.set_head_hardness({.5,.5}) --This sets motor compliance
> b.set_head_command({0,0}) --This moves head to center
> b.set_head_command({1,0})
> b.set_head_command({0,1})
--At this point, feel the stiffness of the head
> b.set_head_hardness({0,0})
--Now feel the stiffness

Play around with some of the other "set" commands.
Hint: try set_larm_hardness, with 3 zeros…

Press “Ctrl-d” when done.

Limb Motion

Challenge!

Make the Darwin raise his hands to the Touchdown sign

Press "Ctrl-d" when done.

Now, reconnect to the DCM ("screen –r dcm") and see the joint values for Darwin's arm. Are they familiar?

Keyframe Motion

Finally, we're going to make our own keyframe motions. First, let's execute some of the predefined keyframe motions.

Change into the "Motion" directory. Execute "cd ../Motion"

Open a file to see the keyframe structure

Execute “less km_OP_StandupFromBack.lua”

-Press “q” when finished glancing at keyframe file

You can see the general structure. Now, we are going to make our own keyframe file (for waving a hand), and play it back.

Keyframe Motion

With the DCM process still running, execute

- “cd ..” to get into the opensource folder
- “lua gen_anim_upperbody.lua”

Move the arms of the robot to the side of the body

Hit enter to appropriately set a keyframe position

Do this a few times

When done, press “g” and “Enter” to generate the file.

Enter ‘mykeyframe’ as the filename and hit enter.

Now run “lua test_keyframe.lua” to test your motion. Enter the appropriate filename.

Penn Engineering
GRASP LABORATORY

Locomotion

Walk basics, Setting walk parameters

Playing with Locomotion

We're going to start telling Darwin how to walk. First, ensure the DCM is running (“screen -r dcm”)

Execute "lua test_walk.lua"

1. Press “8” to make the robot stand up
2. Press “i” to make the robot move forward
3. Press “k” to make it move in place
4. Try side stepping with “h” and “;”
5. Turn left, right with “j”, “l”

Press “7” to make the robot sit down
Press “Ctrl+c” to stop test_walk.

Walk Basics

- Stationary Walking
  -Center of gravity (CoG) always lies in supporting polygon
- Dynamic Walking
  -ZMP always lies in supporting polygon
  -CoG may lie outside of the supporting polygon

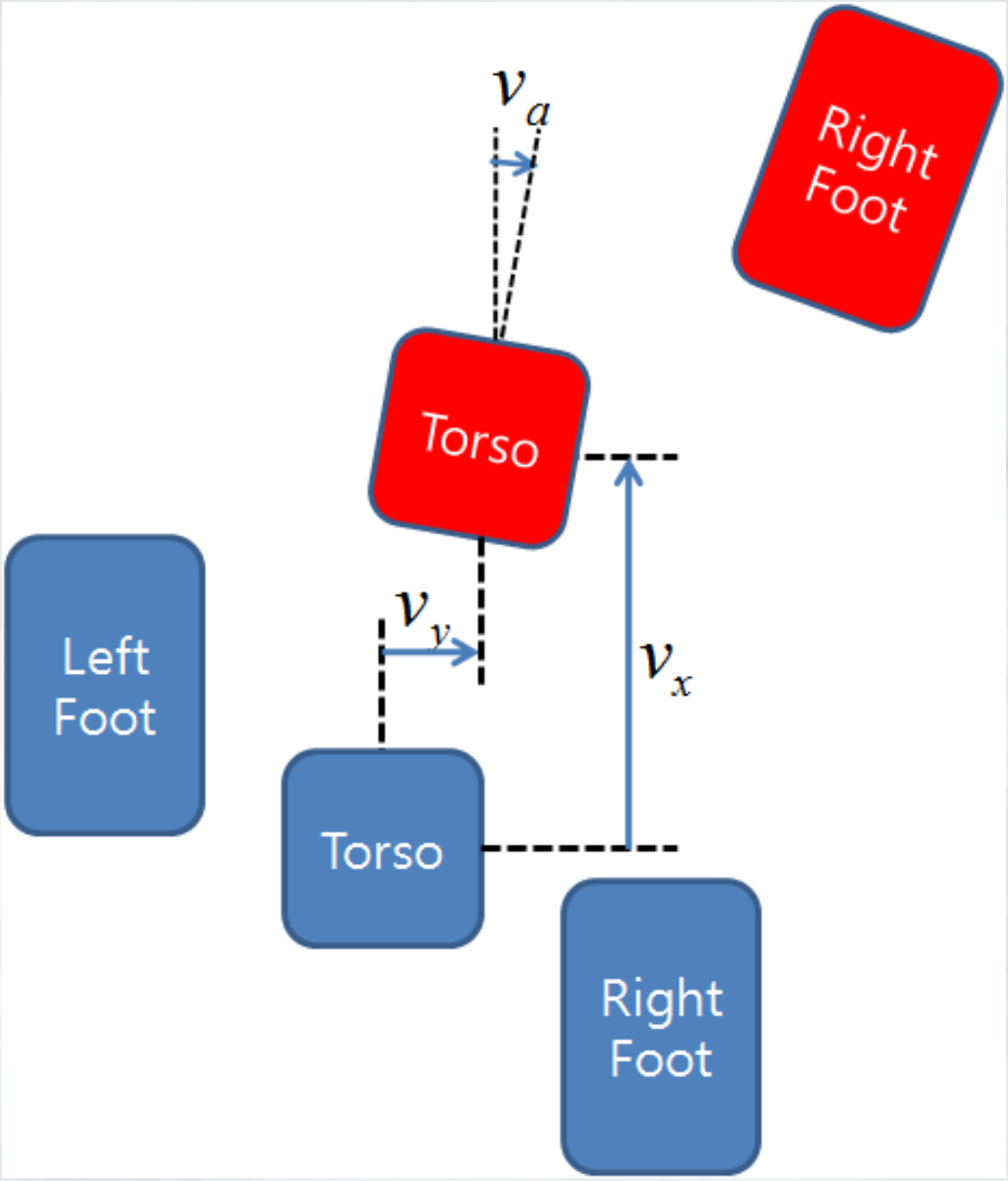
ZMP can be regarded as the dynamic version of CoG. If the ZMP lies in the support polygon during walking, the robot will be dynamically stable.

Our walk algorithm consists of three steps:

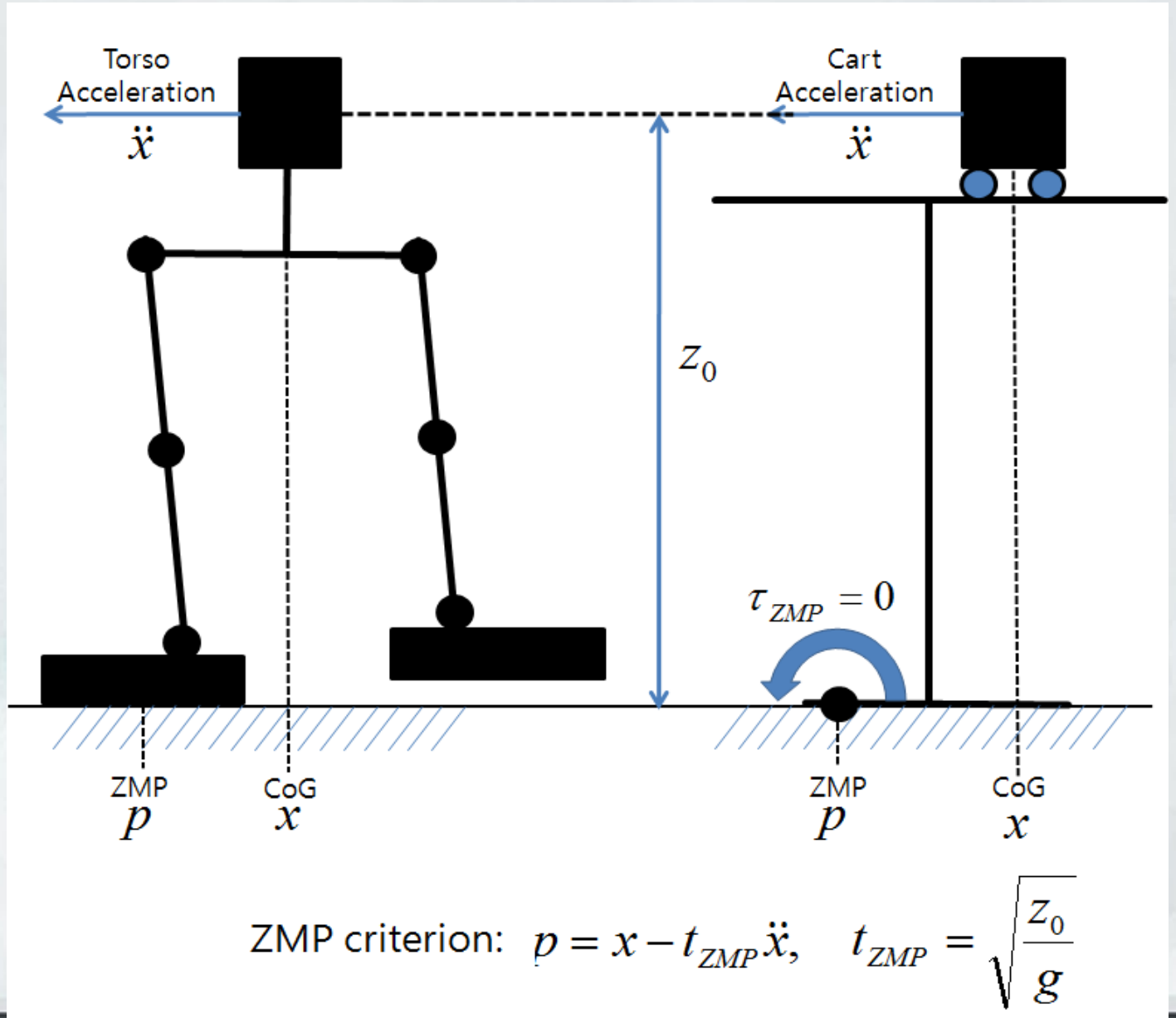
1. Generate foot trajectory according to walk speed
2. Calculate body trajectory to satisfy ZMP criterion
3. Calculate joint angle using inverse kinematics

Foot trajectory generation

-The next torso position is calculated at the start of each walk cycle
-Stepping position is calculated from next torso position
-Omnidirectional walking is possible with specifying separate velocity for x,y,a



ZMP criterion

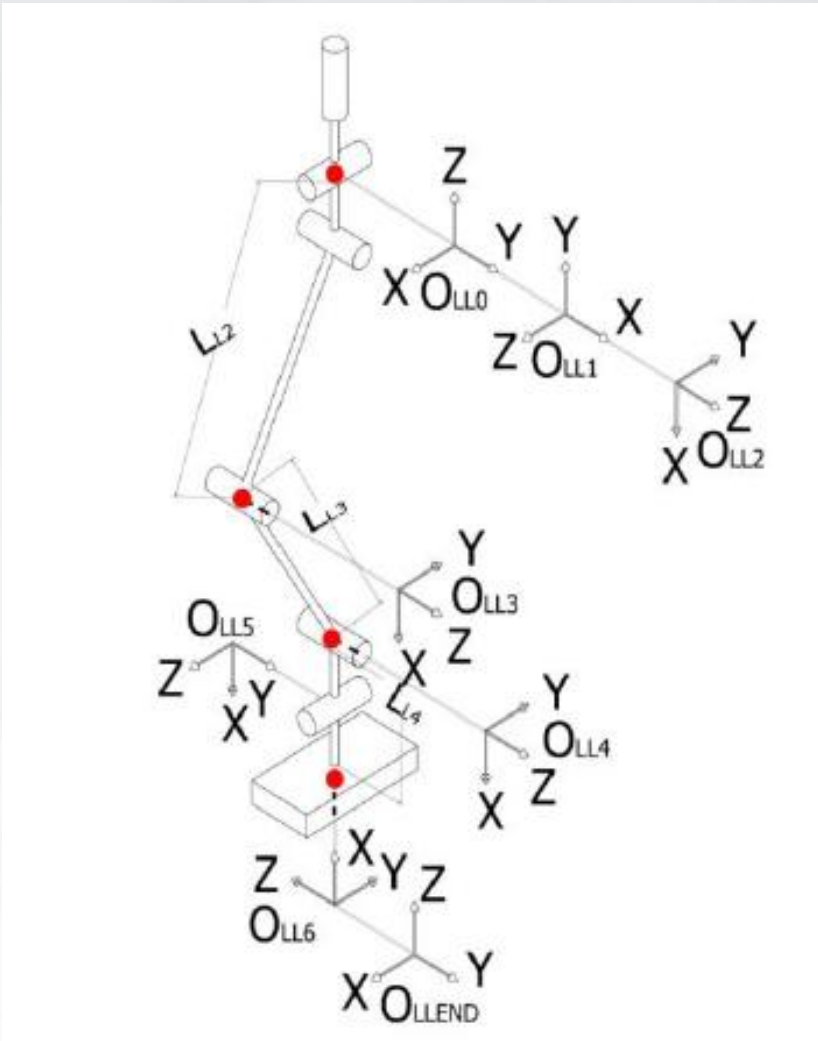


ZMP criterion: $p = x - t_{ZMP}\ddot{x}, \quad t_{ZMP} = \sqrt{\frac{z_0}{g}}$

Joint angle calculation

-We provide front and inverse kinematics algorithm for all limbs to get transform matrix from joint angles and vice versa.

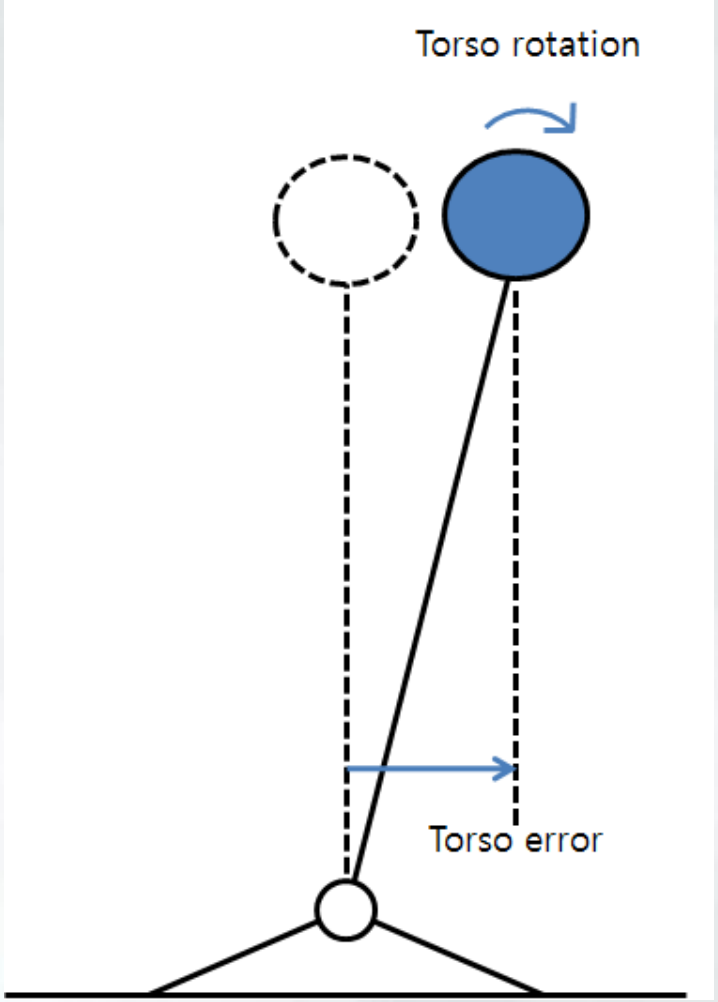
- Limb dimensions are defined in DarwinOPKinematis.h
- Forward and Inverse kinematics defined in DarwinOPKinematics.c



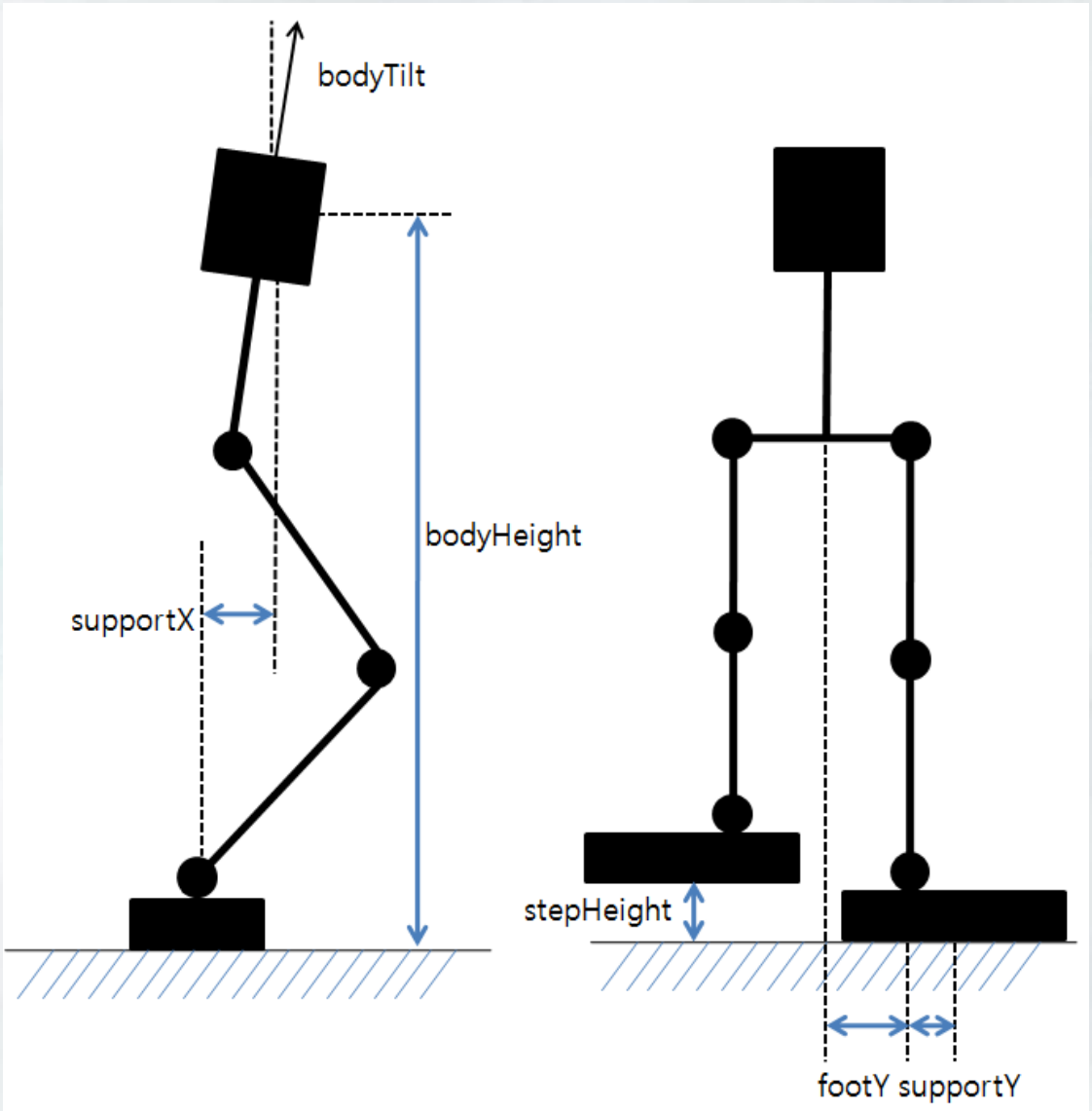
Active stabilization

We provide two ways of stabilizing the walk against external disturbances

- Propriceptory feedback
- Inertial feedback



Walk parameters



Tuning parameters

All walk parameters are defined in Config.lua file

```
walk.bodyHeight = 0.21;
walk.stepHeight = 0.020;
walk.footY = 0.035;
walk.supportX = 0.035;
walk.supportY = 0.00;
walk.bodyTilt= 7*math.pi/180;
....
```

Penn Engineering

GRASP LABORATORY

Changing Walk Parameters

Try modify the walk parameters a bit.

1. Execute “gedit Config/Config.lua”
2. Scroll down to the bottom and you will see walk parameters.
3. Enable commented block of the code by removing “—[[“ and “--]]” lines.
4. Save the file.

Now go up to Player folder (“cd ..”) and run the test_walk.lua again to see how the changed parameter affects the walking.

In case you do something very wrong, there is Config.lua.backup file to help.

Playing with Locomotion

We're going to start telling Darwin how to walk. First, ensure the DCM is running (“screen -r dcm”)

Execute "lua test_walk.lua"

1. Press “8” to make the robot stand up
2. Press “i” to make the robot move forward
3. Press “k” to make it move in place
4. Try side stepping with “h” and “;”
5. Turn left, right with “j”, “l”

Press “7” to make the robot sit down
Press “Ctrl+c” to stop test_walk.

Penn Engineering
GRASP Laboratory

Motion FSM

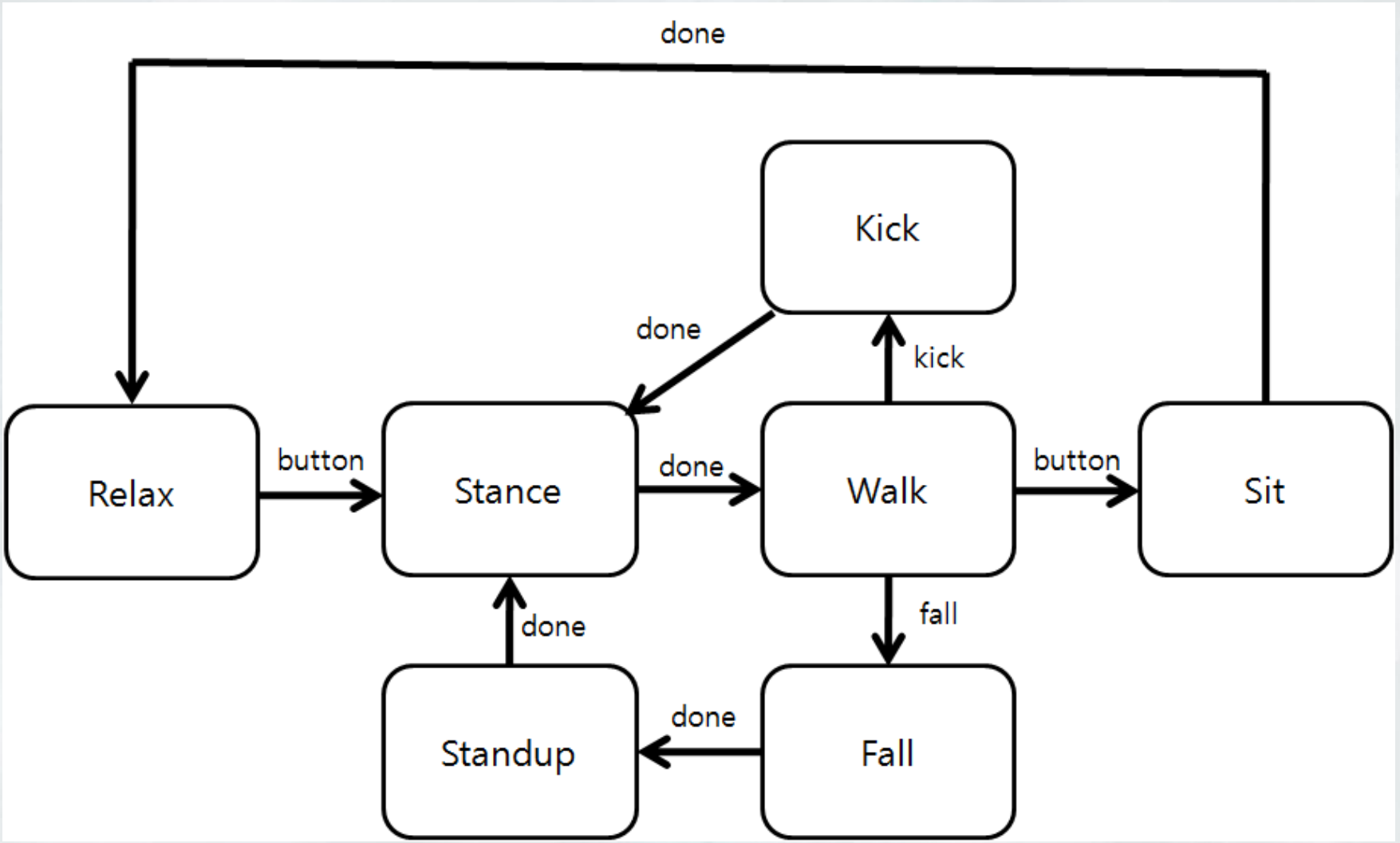
Motion FSM, Managing FSM

What is the motion FSM?

We use a finite state machine named Motion FSM to handle all low level body movements such as:

- Standing up
- Walking
- Sitting down
- Keyframe motion
- Detecting fall
- Automatic standup

Basic Motion FSM



An example of main FSM file

Open the Motion.lua to see an example of an FSM

```
sm = fsm.new(relax);  --Define the FSM, with initial state
sm:add_state(stance); --Add various states
sm:add_state(walk);
sm:add_state(sit);

--Add event transitions
sm:set_transition(relax, "button", stance);
sm:set_transition(stance, "done", walk);
sm:set_transition(walk, "button", sit);
sm:set_transition(sit, "done", relax);
```

Testing motion FSM

You can test the motion FSM as follows;

1. First make the DCM process is running.
2. Execute "lua test_walk.lua" at top folder.

Now the body state machine will wait at relax state.
-You can press the button to make it move into stance, and walk state.
-Try gently setting the robot face down to ground to make it move into falling state, and then standup state.

Press the left button will make the body state move to sit state, and then relax state.

Editing motion FSM

Now let's try modifying the State Machine. What we are going to change the action on a button press.

Open the Motion.lua file with "gedit Motion/Motion.lua"

We see that on a button press from the "relax" state we move into "stance", followed immediately by "walk."

How about just making Darwin-OP stand, not walk?

Editing motion FSM

To do this, we are going to take advantage of the “nullstate” which does nothing on its update routine.

First, require the nullstate near the top of the Motion.lua file

Then, change the transition from stance, on “done” to go to null state. The null state should then transition to relax on “button”

Try to implement this yourself, and when ready, run “lua test_walk.lua” again. Press the button to have Darwin-OP stand, and press the button again to have it sit.

Penn Engineering

GRASP Laboratory

Vision

Device I/O, Processing Library

Device I/O

What you need to know:

- Darwin-OP has a USB webcam for computer vision.
- The webcam communicates via UVC to Linux
- Darwin-OP's software contains Video4Linux2 drivers to sample from the camera at 30Hz
- For debugging purposes, each frame is stored in a shared memory file

Processing library

The Vision system is divided into two areas. First, C++ routines perform raw calculations on images. Second, a Lua FSM takes these calculations and updates the global ball location

The C++ routines are located in "~/Desktop/opensource/Lib/ImageProc"

The Lua FSM is located, as suspected, in "~/Desktop/opensource/Player/Vision"

Penn Engineering

GRASP LABORATORY

Processing library

At the heart of the image processing library is the Lookup Table, or LUT for short. Every pixel in the image is assigned to a discrete color label, such as “Orange“, “Green“, “White" etc.

After grabbing an image, we find the largest connected components of ball pixels.

In order to reduce noise, we partition the image into 4x4 pixel blocks.

Behavioral FSMs and main control code

What are behavioral FSMs?

Behavioral FSMs are finite state machines that govern high level behavior of the robot.

- Head FSM
  - looking around to find the ball
  - Track the found ball
- Body FSM
  - Search the area until ball is found
  - Approach the ball until the ball is close enough to kick
  - Kick the ball

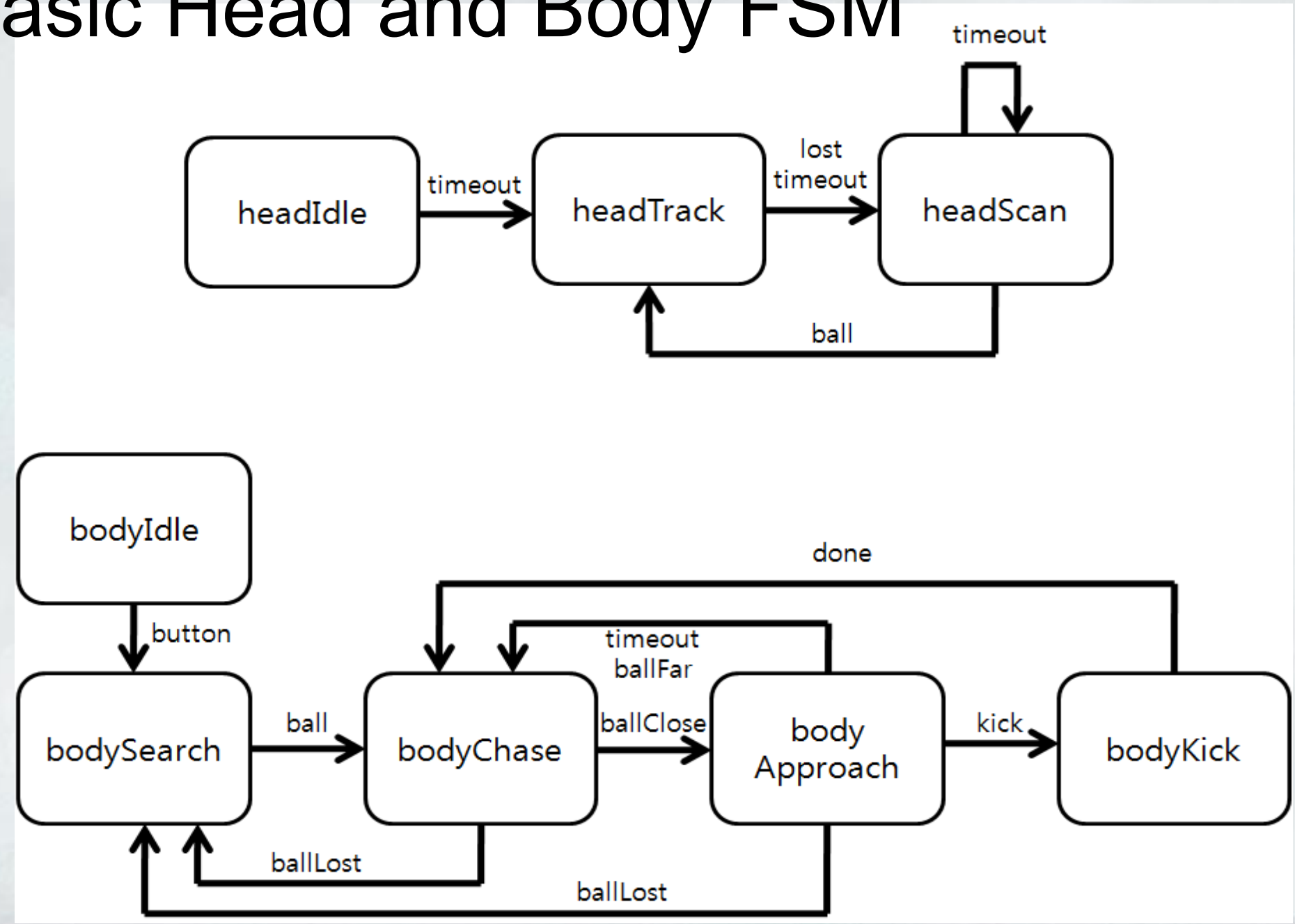
The main control code

Main controller code simply initializes a number of behavioral FSM and continuously updates them.

```
Motion.entry();
BodyFSM.entry();
HeadFSM.entry();
Vision.entry();

while 1 do
  Motion.update()
  BodyFSM.update()
  HeadFSM.update()
  Vision.update();
end
```

Basic Head and Body FSM

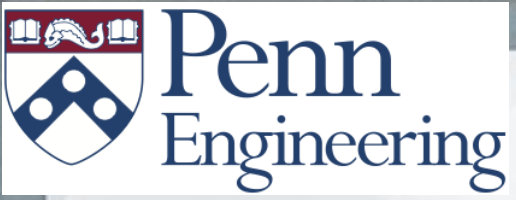


Testing main control code

Now you can run the full player code with all FSM running.

1. First make the DCM process is running.
2. Execute "lua player.lua" at top folder.
3. Press left button to make the robot stand up.

Now Darwin-OP will find a ball, approach the ball, and kick it.
Can you see which state the head and body FSM is in?
After playing for a while, you can press the left button to make the robot sit down.



GRASP
LABORATORY

Thanks!

Testing Vision

In the “Player” directory, run “lua test_vision.lua” to have DarwinOP track the ball.

Press “8” to have DarwinOP stand up
Press “i”,”j”,”l”,”,” to move the head around
Press “1” to toggle tracking of the ball

When done, press “7” to have DarwinOP sit down
Press “Ctrl-c” to end the process