Integrating Johnny 5 with LTLMoP

Author: Deng Deng

NetID: dd526

12/18/2013

Table of Contents

1. Overview

This project aims to control Johnny 5 robot using LTLMoP. Primarily, this entails creating handlers that will pass the commands from LTLMoP to Johnny 5. Since LTLMoP takes in task specification in structured English, it must be interpreted into robot-specific commands for control purpose. For example, LTLMoP may pass a command of "go to region r2" onto the motion controller, but the robot needs to receive a velocity in local coordinates to move. Controlling of robot is realized using a set of handlers in LTLMoP, which will be elaborated in subsequent sections. Communication with the robot will be via XBee wireless modules, which are easy to setup and capable for this project.

2. LTLMoP Handlers

There are seven required handlers for each robot in LTLMoP: init, pose, motion control, drive, locomotion, sensor, and actuator. Each handler is briefly described below:

Init: Set up connections to the robot and share necessary data with the other handlers
Pose: Extract position and orientation information from Vicon
Motion Control: Given the region to move to next, output a global velocity vector in that direction
Drive: Take global velocity vector and translate into values relevant to the robot
Locomotion: Take relevant movement-related values and command the robot to move
Sensor: Call sensor function on the robot and translate return values into Boolean propositions
Actuator: Make the robot do a custom movement or other change of state

In this project, Init, Locomotion, Sensor and Actuator handlers are written for Johnny 5. Existing viconPose, vectorController and differentialDrive handlers are used for robot localization and path planning.

3. Johnny 5

3.1 Overview

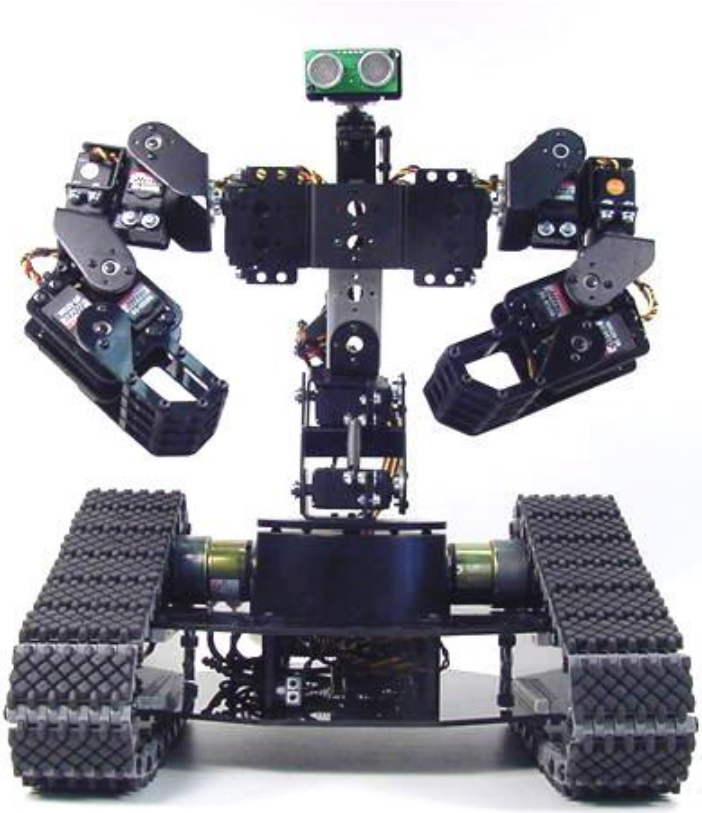

Figure 1. Johnny 5 picture

The Lynxmotion Johnny 5 is used in this project. It has a dimension of 10 x 11 x 14.25 inch and can move up to 24 inch/sec. The robot is made from Servo Erector Set aluminum brackets, custom injection molded components, and ultra-tough laser-cut Lexan structural components. Specifically, Johnny 5 is a multi-servo driven robot with polypropylene and rubber tracks. It is designed for indoor or outdoor use and performs well on many different surfaces. Despite its compact size, Johnny 5 is capable of accomplishing various tasks. It can be controlled by Lynxmotion Sequencer Program, either with cable or with wireless connection. It can also accept commands sent directly to its servo controller in a pre-determined format.

Johnny 5 is composed of 14 servos and 2 motors as listed below:

- 8 x HS-645MG servos
- 3 x HS-475HB / HS-485HB servos
- 3 x HS-422 servos
- 2 x 12vdc 50:1 gear head motors with Sabertooth 2 x 5 motor controller

It is equipped with two batteries, one 6V battery pack to control all servos and one 12V battery pack for motors. Detailed specifications of Johnny 5 is given below:

- Length = 10.00"
- Width = 11.00"
- Height (Overall) = 14.25"
- Height (Base and torso) = 11.75"
- Ground Clearance = 1.00"
- Deck Height = 3.00"
- Deck Dimensions = 7.25" W x 8.125" D
- Weight = 6.25 lbs (no batteries)
- Speed = 24.0 in/sec

3.2 SSC-32 Servo Controller

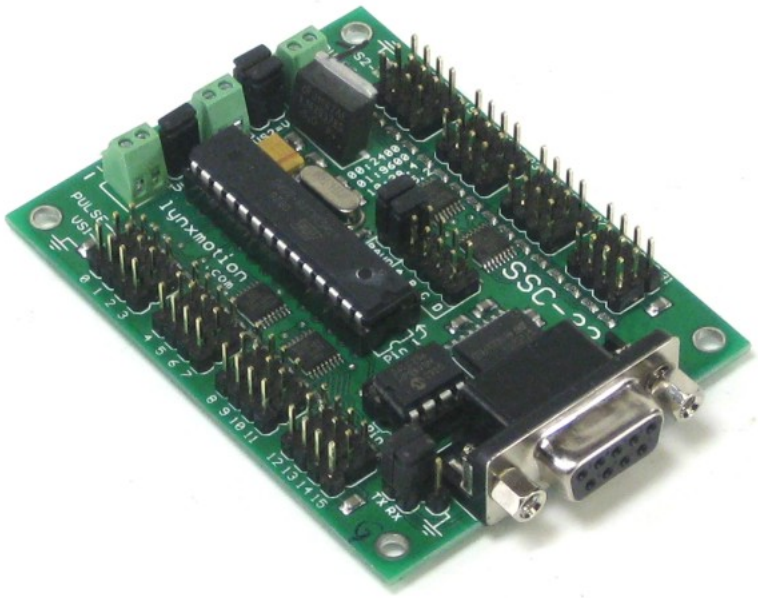

Figure 2. SSC-32 servo controller

The SSC-32 servo controller is used to control both servos and motors in Johnny 5. It is a preassembled serial servo controller with accurate positioning and smooth movements. The motion control can be immediate response, speed controlled, timed motion, or a combination of them all. A unique feature of the SSC-32 is the "Group Move" feature that allows any combination of servos to begin and end motion at the same time, even if servos have to move different distances. There is also bidirectional communication with Query commands, meaning that the SSC-32 is able to provide feedback to the host computer.

Since Johnny 5 has a total of 16 servos (including 2 motors), pins numbered 0-15 are used on SSC-32 to control the robot. Pin numbers with corresponding servo movements in Johnny 5 are listed below:

#0 base rotates

#1 lower waist leans forwards/backwards

#2 upper waist leans forwards/backwards

#3 left shoulder rotates

#4 left shoulder moves inwards/outwards

#5 left arm rotates

#6 left arm moves inwards/outwards

#7 left hand grasps

#8 right shoulder rotates

#9 right shoulder moves inwards/outwards

#10 right arm rotates

#11 right arm moves inwards/outwards

#12 left hand grasps

#13 head rotates

#14 track moves forwards/backwards: static@1500; forward@<1500; backwards@>1500

#15 track rotates: static@1500; Clockwise@>1500; Counterclockwise@<1500

To control the servos, a predetermined syntax is used for all commands. The generalized format of a command looks like:

 # + "Servo Number" + " " + P + "Servo Value" + " " + T + "Time in ms" + \r

Where:

Servo Number – servo pin number, from 0-15

Servo Value – servo value between 500 and 2500

Time – time during which servo completes its movement in ms

\r – required format that specifies carriage return to end the command

An example of command is given below:

#0 P1518 T1000 \r

It means to set servo #0 value to 1518 in 1000ms.

3.3 Wireless Communication

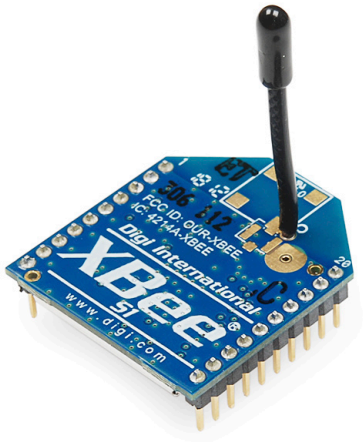

Figure 3. XBee 1mW wire antenna - Series 1

Two XBee Series 1 modules are used for wireless communication in this project due to their easy setup and good performance. Both modules are coupled with XBee Explorer, one is connected to the computer end, another is attached on Johnny 5 and connected to SSC-32 servo controller.

The schematic for connecting XBee Explorer and SSC-32 is shown in Figure 4.

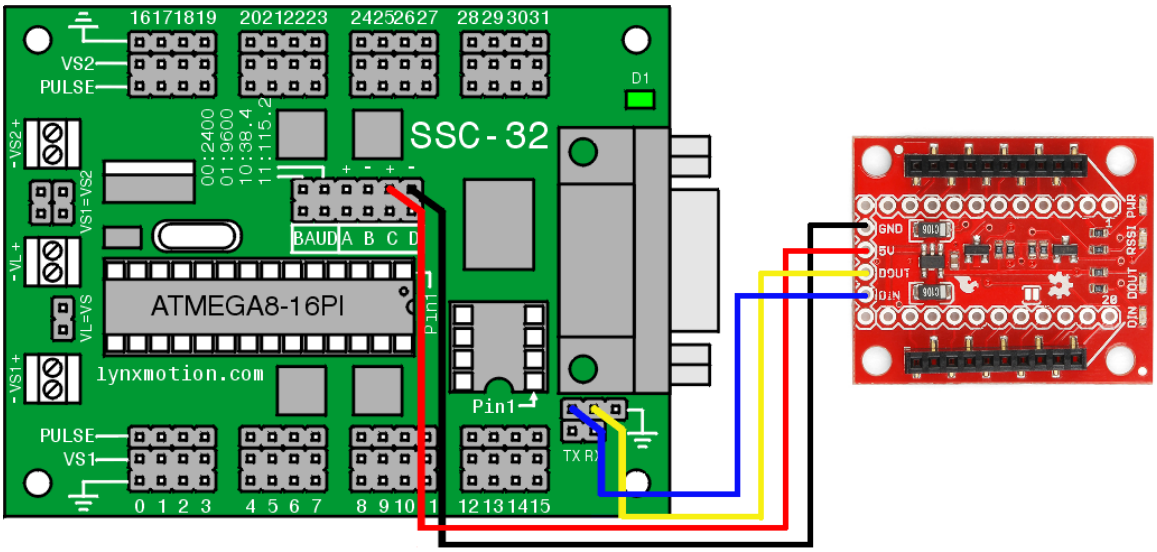

Figure 4. XBee Explorer connection with SSC-32

X-CTU is used to configure XBee modules. Both modules should be set to share the same channel and Pan ID, baud rate is set to 115200 to match that in SSC-32. Serial numbers of both XBee modules can be found using X-CTU, in this project, broadcasting and recipient XBee serial number are 13A20040A5C737 and 13A2004033D787 respectively. During configuration with broadcasting XBee, the recipient XBee serial number needs to be recorded as destination address and vice versa in order to communicate between modules.

4. Lynxmotion Sequencer

Lynxmotion Visual Sequencer is a Windows program for controlling anything built using up to 32 servos with the SSC-32. It comes with the robot and the install CD is stored together with Johnny 5 inside the box.

Sequencer allows initial calibration of all servos on Johnny 5 to neutral positions and saves that as a .cfg file. On every startup, this configuration file will be loaded to the robot as reference for subsequent commands.

Another feature of Sequencer is that you can build a project composed of a sequence of servo movements. Within project, the main screen allows you to add servo control boxes, and position them on a grid. This visual representation of the robot makes it easier to position the servos for each sequence. A snap shot of Johnny 5 project is shown below.

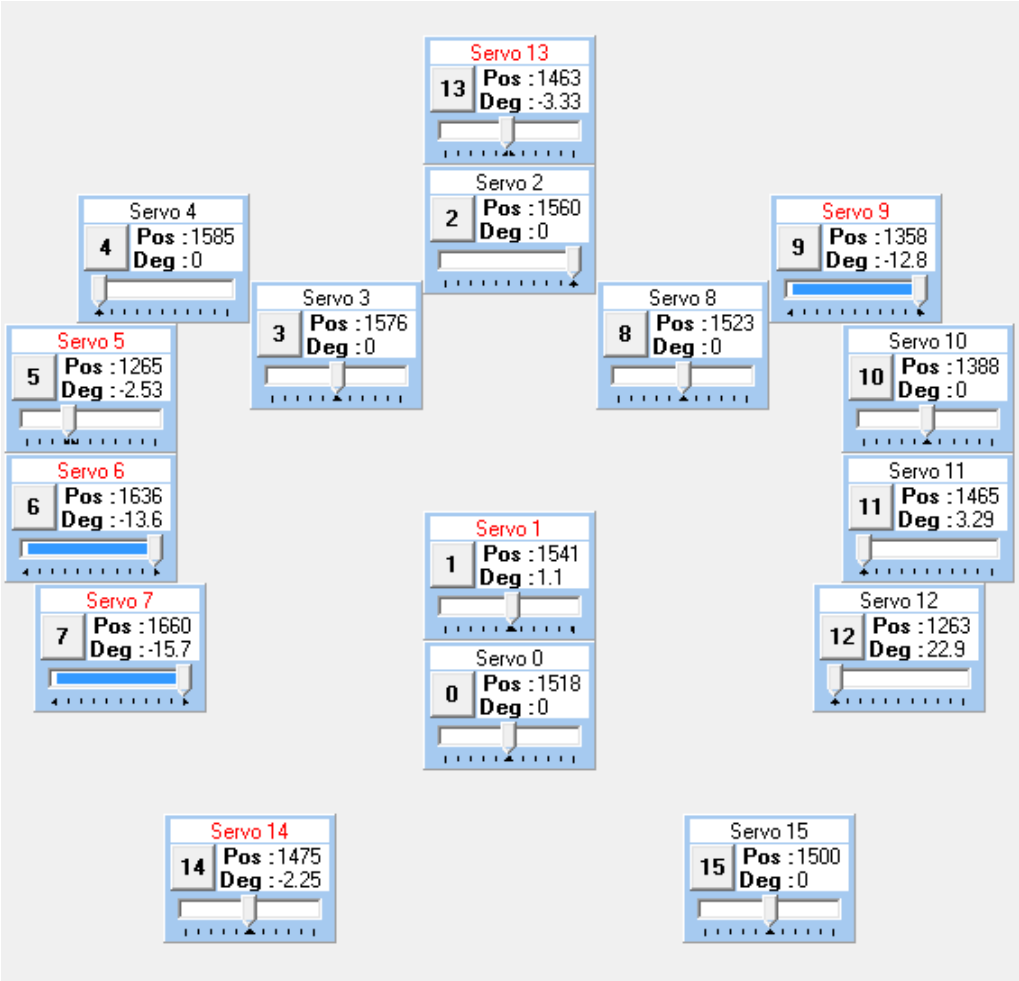

Figure 5. Interface of Lynxmotion Visual Sequencer

As can be seen in the figure, moving each servo to the desired position composes a movement for the robot. Each servo will therefore have a specified degree and value, as well as time constant for one step movement. A sequence can have one or many steps, and a Sequencer project consists of one or more sequences. After making up a project, it can be exported in Sequencer to two files, one is .shp file and the other is .csv file, both share the same project name. Although both files are needed when we want to import the project in Sequencer, only the .csv file is concerned in LTLMoP.

In this project, Sequencer will be used to generate a sequence of robot movements. The corresponding .csv file will then be parsed inside LTLMoP handler to output correct servo commands via XBee. It is worth noting that for convenience, all .csv files are put under folder SequencerFiles inside the Johnny5 folder, and the .cfg file is saved under folder ConfigFile, this gives a more organized way to visualize handlers and Sequencer files. Both folder names and paths can be modified in Johnny5Init handler and Johnny5Actuator handler respectively.

5. Handlers

As mentioned above, LTLMoP uses handlers to control the robot. Since Johnny 5 moves on its rotating tracks, which resembles normal wheel drive system, it can use the existing differentialDrive module to calculate desired velocity and angular velocity along the path. Another two existing handlers, viconPose and vectorController, are used to locate Johnny 5 using Vicon data and calculate path vectors from region to region respectively. Apart from this, four handles, namely Init, Locomotion, Sensor and Actuator, need to be written and will be illustrated below.

5.1 Johnn5Init

Johnny5Init handler is called once LTLMoP starts up. In summary, three actions are taken at this stage, they are serial communication setup, default servo positions read/set and shared data setup.

Serial setup basically specifies comport and baud rate, where comport is chosen in LTLMoP GUI and baud rate is set to 115200. A servo configuration file from Sequencer program is saved under folder ConfigFile. It indicates 5 values for each servo, namely, neutral servo value, minimum servo value, corresponding minimum servo degree, maximum servo value, and corresponding maximum servo degree. The Init handler then reads the file and saves as an array, which is used for sending out serial commands to setup default servo positions. Both serial connection and servo configurations are then shared with other handlers.

5.2 Johnny5LocomotionCommand

This handler receives velocity and angular velocity value generated from differentialDrive and sends out desired servo value to the robot. A gain of 1000 for both velocity and angular velocity are chosen from experiment to amplify given value to servo position perceivable in Johnny 5. As mentioned above, neutral servo positions for track moving and rotating are 1500, so final servo output for both is:

*1500-Velocity Gain*Velocity Value (Angular Velocity Gain*Angular Velocity Value)*

However, a small velocity value generated from Drive handler will not be able to move the robot due to friction. On the other hand, it should be noted that although valid servo value ranges from 500 to 2500 theoretically as mentioned above, this is not the case for servo #14 and #15. It is found in tests that both servos accept a maximum of +/- 500 values to function, servo value beyond that will cause motors to stop.

Therefore, a minimum servo threshold and a maximum boundary are introduced to ensure servo value is within desired range. For velocity, it is set to 100 and 300 respectively. For rotation, the servo value is bounded to 100-500, considering a larger value delivers more satisfied turning speed.

5.3 Johnny5Sensor

In the final demonstration of this project, Johnny 5 will try to find me wearing the helmet with Vicon balls on. In the Sensor handler, a Boolean method is used to indicate if the helmet is found within certain range of the robot position. Since both Johnny 5 and the helmet are modeled in Vicon, Vicon real time data can be used to locate the objects with Pose handler. Hence, use simple math can calculate the distance between robot and helmet.

5.4 Johnny5Actuator

The Actuator handler is responsible for performing different tasks, which consist of a sequence of servo movements. It first looks at shared data and from which establishes a serial connection, and saves the default servo configuration data.

A runSequencer method is written to parse .csv file generated from Sequencer program to serial commands with corresponding servo positions and timing. A sample format of .csv file and its derived commands will be shown below. Since we are only using 16 servos in this project, the first table shows desired servo value in degree for servo 0-15, the second table indicates corresponding servo timing in ms.

PROJECT	SEQUENCE	STEP	PIN0	PIN1	PIN2	PIN3	PIN4	PIN5	PIN6	PIN7	PIN8	PIN9	PIN10	PIN11	PIN12	PIN13	PIN14	PIN15
TakeBow	1	1	-29.966	0	0	0	0	0	0	0	0	0	0	0.094	0.097	0	0	0
TakeBow	1	2	-29.966	-30	0	45.049	0	0	-59.866	0	45	0	0	59.969	1.257	0	0	0
TakeBow	1	3	-29.966	0	0	0	0	0	0	0	0	0	0	0.094	0.097	0	0	0
TakeBow	1	4	29.967	0	0	0	0	0	0	0	0	0	0	0.094	0.097	0	0	0
TakeBow	1	5	29.967	-30	0	-30.032	0	0	-45.05	0	-45	0	0	59.969	0.097	0	0	0
TakeBow	1	6	29.967	0	0	0	0	0	0	0	0	0	0	0.094	0.097	0	0	0
TakeBow	1	7	0	0	0	0	0	0	0	0	0	0	0	0.094	0.097	0	0	0

Figure 6. Format of a .csv file column 0-18

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Figure 7. Format of a .csv file column 35-50

Let's first look at step 1, only servo 0 receives a substantial degree of approximately 30. As the shared configuration data includes minimum servo value MinV, corresponding minimum servo degree MinD, maximum servo value MaxV, and corresponding maximum servo degree MaxD, the required servo value output is calculated from:

$$\mathit{Servo\ Value} = \mathit{int}(\mathit{MinV} + (\mathit{Degree}-\mathit{MinD})/(\mathit{MaxD}-\mathit{MinD})*(\mathit{MaxV}-\mathit{MinV}))$$

Given MinV = 635, MinD = -90, MaxV = 2415, MaxD = 90 for servo 0, its servo position is therefore:

$$\mathit{int}(635 + (-29.966-(-90))/(90-(-90))*(2425-635)) = 1232$$

Servo timing can be directly read from table, which is 1000 for servo 0.
Thus, servo command for servo 0 is:

#0 P1232 T1000 \r

All other servos will use the same formula to generate the command, 0 degree is therefore basically the neutral servo value.
In between each step, the method sleep for an amount that is the maximum servo timing in such step, to ensure sequential servo movements. For example, between serial commands from step 1 and step 2, a time.sleep(1000) line is inserted.

The runSequencer method makes the handler versatile and easy to use. For example, we have exported a Sequencer project “TakeBow.csv” and put it in folder “SequencerFiles”. In Johnny5Actuator, we can simply add a new method takeBow and call:

```
self.runSequencer('TakeBow.csv')
```

It will first look up the file in the destined folder and send out all appropriate servo commands, the robot should follow the exact sequence of movements designed in Sequencer.

6. LTLMoP Specifications

In the final project demonstration, a custom-made region map is shown below. To start, Johnny 5 will be placed inside one of the region. The robot will do a “takeBow” action before traversing around the map to find me with a Vicon helmet. As long as Johnny 5 finds me within a certain distance, it stops and does “highFive”. “findMe” is the boolean method in Johnn5Sensor handler doing the distance calculation. The corresponding LTLMoP specification is also presented.

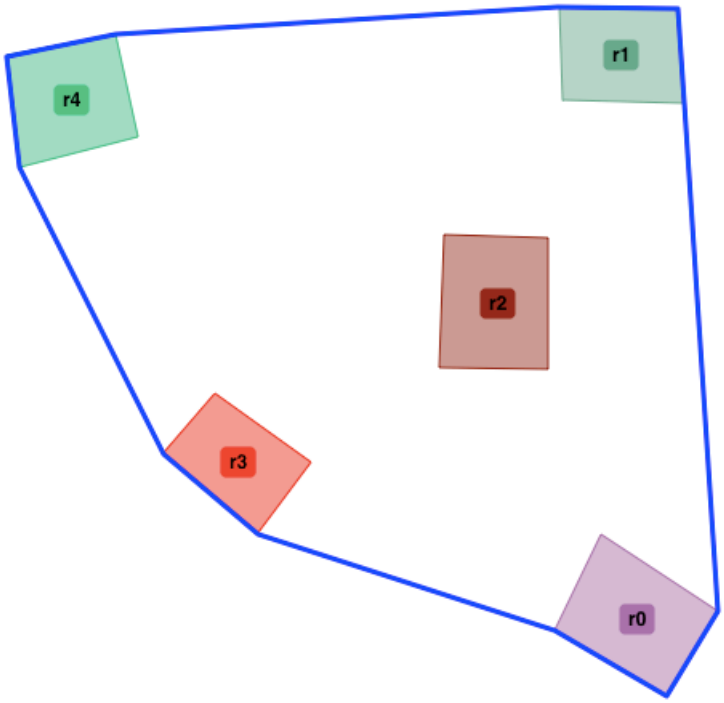

Figure 8. Region map in LTLMoP

```
1  group Map is r0, r1, r2, r3, r4
2  robot starts in any Map with false
3
4  #do takeBow
5  visit all Map
6
7  if you are sensing findMe then stay there
8  do highFive if and only if you are sensing findMe
9  infinitely often not findMe
```

Figure 9. LTLMoP specifications

7. Recommendations For Future Work

Currently Johnny 5 uses external resource like Vicon to maintain its states such as localization and path planning. It will be beneficial if the robot can keep track of its own states in the case of lost communication. Therefore, sensors such as Gyroscope, Ultrasonic and Infra-red can be added to Johnny 5 for position inference, obstacle detection and avoidance etc.

Although SSC-32 servo controller can read directly from analog input, adding an intermediate microcontroller such as Arduino to handle sensor data is recommended. Arduino is compatible with SSC-32 servo controller and it is capable of intensive onboard computation, which helps future development.

8. Troubleshooting

1. If Lynxmotion Visual Sequencer can't find SSC-32 card at startup…

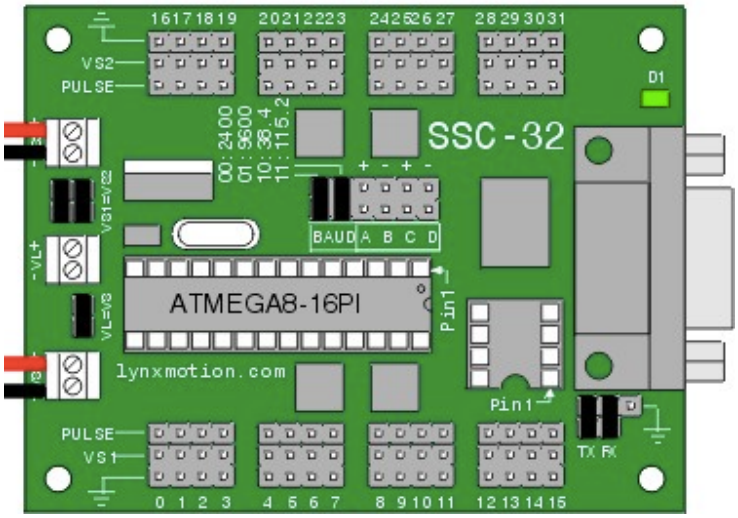

Figure 10. SSC-32 servo controller with default attached jumpers

a. If using DB9 cable for connection…

i. Default Baud rate of SSC-32 is 115200, check if both jumpers are attached in BAUD pins. Jumpers' position and corresponding Baud rate are shown below:

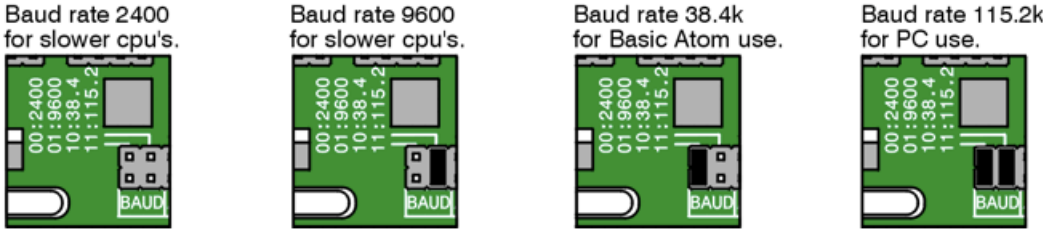

Figure 11. BAUD pin jumpers' positions and rate

ii. Check COM port on the computer end is set to the same baud rate.

iii. Make sure both jumpers on TX and RX pins are attached, which enable DB9 port.

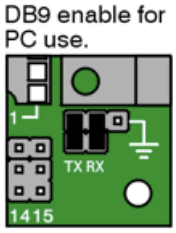

Figure 12. Jumpers' position to enable DB9 cable

   b. If using XBee wireless module…
      i. Check both broadcasting XBee and recipient XBee share the same Baud rate, channel and Pan ID.
      ii. Check in broadcasting XBee configuration, if the recipient XBee serial number has been recorded as destination address, vice versa.
2. If in LTLMoP robot configuration, a handler is missing…
   a. There is error in the handler code, check terminal for debug information.
3. If in LTLMoP simulation, there is no robot pose from Vicon…
   a. Make sure Vicon cameras and software are all started up.
   b. Made sure your computer is connected to ASL wifi, not RedRover.
4. If in LTLMoP simulation, it shows connecting followed by a timeout…
   a. Made sure your computer is connected to ASL wifi, not RedRover.
5. If robot doesn't go forward to a desired region in LTLMoP…
   a. If robot keeps going out of bound without rotating…
      i. In Johnny5LocomotionCommand handler, the velocity value input to servo should be in opposite direction.
   b. If robot rotates and goes backwards towards destination…
      i. In Johnny5LocomotionCommand handler, the angular velocity input to servo should be in opposite direction, assuming the velocity is already in right direction.