Johnny 5: Paparazzi

Amol Akalkotkar, Sandeep Bajwa, Divya Bhat, Danqian Chen, Rishi Danju, Santosh Khadkha, Prashant Kumawat, Dhairya Patel, Nehal Patel, Yogesh Sanghi, Dhrubojyoti Bhattacharjee,Sashidhar Reddy,Nishant Gemawat, Bhumika Kachhadia, Pratishara Maharajan, Amit Mahindrakar, Sonell Raman, Biyana Shreshtha, Laxman Thapaa.

Coordinator: Dr. Stefan Andrei

Lamar University
Department of Computer Science
Beaumont, Texas

Abstract: Embedded systems are designed to do some specific tasks often with real-time computing constraints. The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. The hardware and software components are synchronized using these instructions. This paper describes how we implemented and examined the development of an embedded system, namely Johnny 5. It is a robot manufactured by Lynxmotion, Inc., a U.S. company specialized in designing a great variety of programmable embedded systems, especially robots. A powerful Bot Board/SSC-32/BA28 and a PS2 Controller are attached to Johnny 5 to simulate the problem. The application chosen for the robot is called "Paparazzi". Its mission is to hold a recording camera in its hands, avoiding obstacles, and then return back to its initial point following the same path. While working on this project, our team experienced certain challenges during implementation phase. Our project was of a much higher complexity than the earlier projects like Mini-hexapod, Mini Rover in terms of movements and functionality.

1. Introduction

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as a component of a complete device often including hardware and mechanical parts. Embedded systems are controlled by one or more main processing cores that are typically either a microcontroller or a digital signal processor (DSP). The key characteristic is however being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites.

The Johnny 5 robot is made from Servo Erector Set aluminum brackets, custom injection molded components, and ultra-tough laser-cut Lexan structural components [3]. The torso is fully articulated utilizing 8 x HS-645MG, 3 x HS-475HB / HS-485HB, and 3 x HS-422 servos, and a SSC-32 servo controller. By utilizing heavy duty polypropylene and rubber tracks with durable ABS molded sprockets the robot has excellent traction. It includes two 12vdc 50:1 gear head motors and the Sabertooth 2 x 5 motor controller. It can be controlled from a PC using a DB9 cable. This robot is provided by Lynxmotion, Inc.. It seems that Johnny 5 along with AH3-R robot were used in the laboratory of the new 'Knight Rider' movie.

Lynxmotion, Inc., was founded in 1995 due to a fascination with robotics and a frustration with the available robot-based kits [2]. The company has a large selection of intelligently designed, precision engineered robot kits and components. The company has many innovative solutions to common robotics problems. In addition to many products, the company provides valuable information on doing hobby robotics. Technical support is also provided by the assembly guides on their website [2] and other communication media (email, phone).

2. Related Works with Johnny 5 Robot

We only enumerate two similar robots from Lynxmotion: the A4WD1 and the Sumo robot.

The A4WD1 v2 Robot: The A4WD1 v2 Robot has rolling chassis, RC truck tires and wheels allowing the robot to get excellent traction. The aluminum 4WD1 Robot Kit is a robust, modifiable, and expandable chassis for RC or autonomous robot experimentation. It includes Bot Board, Basic Atom Pro and three GPD2D12 sensors for obstacle detection and avoidance [2].

The Sumo Robot: The Sumo Robot is controlled with dual motor controller and RC set for remote control which can be used for autonomous operation. It includes the chassis made from ultra-tough laser-cut structural components. The basic Stamp 2 is the controller with a basis StampBoard as its motherboard. It uses a sensor (that is, Devantec SRF04 Ultra Sonic Finder) for detecting the distant objects [2].

3. Background

Similar to all the other embedded systems, the Johnny 5 has two basic components: hardware and software components. The hardware component consists of a SSC-32/BA28 controller and servos assembled together to perform the moves specified by the software component. The software component is programmable, and is responsible for controlling the movement of the Johnny 5 robot [3].

3.1. Software Development Environment

The BasicMicro-IDE is a Windows-based Integrated Development Environment (IDE) for micro-controllers (See Figure 3.1.1). Originally designed for use with Micro-C, the Micro-IDE is fully user-configurable to convert command line compilers, assemblers and utilities into Windows applications. It is a Multi-file Editor with tabs to create and modify C or Assembly source code. It has a Built-in Loader to download programs to target the microcontroller board following build. It also includes Built-in Terminal to interact with the target microcontroller board. In addition, it has many features such as customizable settings

and command-line options for building projects, ASCII chart and so on. The BasicMicro IDE supports many external toolkits such as C compilers, BASIC compiler and assemblers.

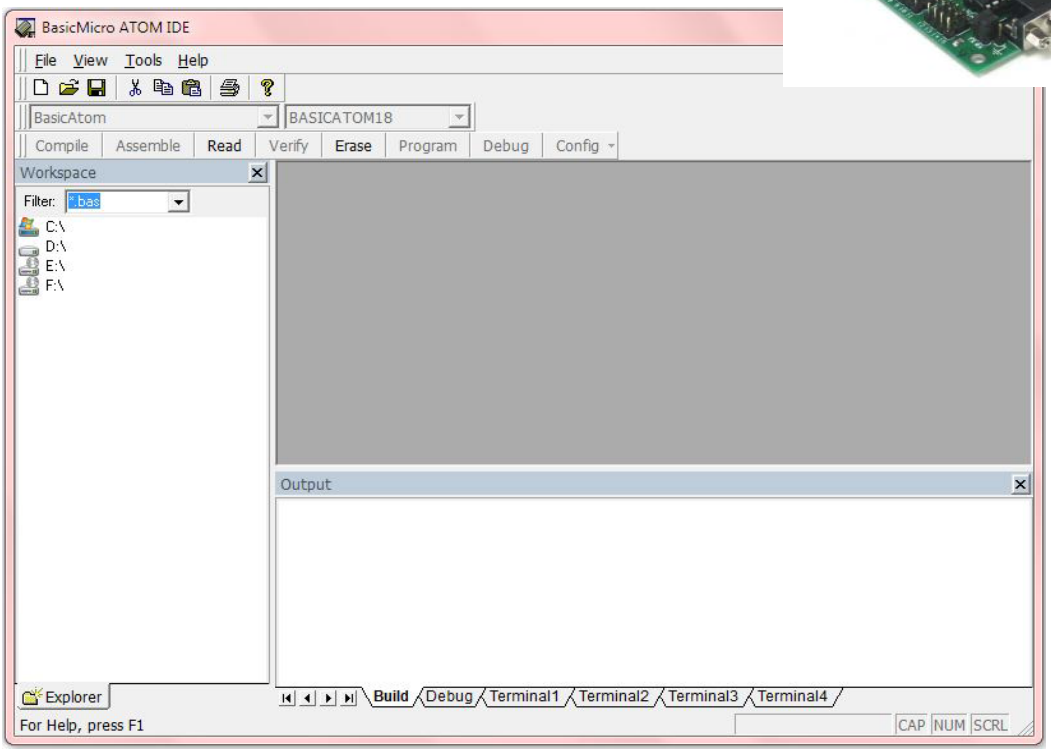


Figure 3.1.1 BasicMicro Atom IDE v5.3.1.0

3.2. Hardware Specification

The Johnny 5 includes a powerful Board/SSC-32/BA28 and a PS2 controller [3]. The hardware contains the Johnny 5 Body Kit and many HS-422 (57 oz. in.) Standard Servos. A powerful Basic Atom 28 Pin is faster and has more memory than a BS2. This chip is relatively easy to program and reliable to use. It can be plugged into the Bot Board for complete access to all of the I/O pins. It is BS2 Pin Compatible and includes O'Scope and In Circuit Debugger (ICD).

The SSC-32 Servo Controller is the best servo controller available which has 32 channels of 1uS resolution servo control. It has features such as Bidirectional communication with Query

commands, Synchronized or "Group" moves and 12 built in Servo Hexapod Gait Sequencer, MiniSSC-II emulation.

The Sabertooth 2X5 R/C is a Regenerative Dual Channel Motor Controller.

The DB9 Serial Data Cable is used to connect to a SSC-32 servo controller with the computer in order to configure the Johnny 5.

These laser-cut Lexan parts and nylon standoffs make a nice Biped hand. The parts can make either a left or right hand. The servos are not included. It can be used for a Humanoid Biped, or any project that needs to get a grip.

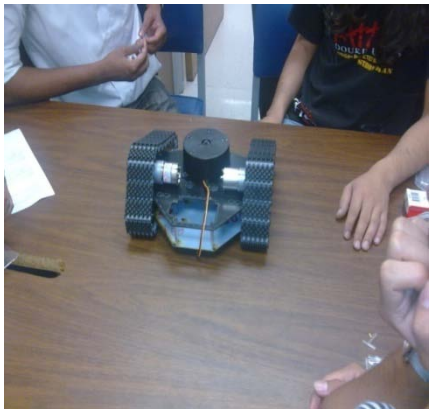
High quality black anodized brushed aluminum bracket makes easy work of mounting any of the popular sensors. The universal mounting holes allow attaching the SRF-04, SRF-05, SRF-08, Ping))) or the Sharp GP2 series of IR sensors to a servo. The bracket can also be used to static mount the sensors to a robot body. It includes hardware to mount a single sensor to the bracket, and to mount the bracket to a servo.

4. Design and Implementation

The specifications defined for this project constitutes of both hardware and software components. The design for the hardware component primarily involves setting up the Johnny 5 [3]. There are five main parts in assembling Johnny 5 which are as follows:

4.1 **Assembling the Tracks:**

4.2 **Mounting the Tri-Track Chassis:**



The below table contains the Sabertooth connection information, in particular the switch settings we used for Johnny 5 application.

Sabertooth Connections		Sabertooth Switch Settings	
M1A	Robot's right motor, red wire	1	Independent Control
M1B	Robot's right motor, yellow wire	2	Disable Exponential
M2A	Robot's left motor, red wire	3	Non-lithium mode
M2B	Robot's left motor, yellow wire	4	R/C Flip Mode
B+	Battery (+), red wire	5	Enable Auto-Calibrate

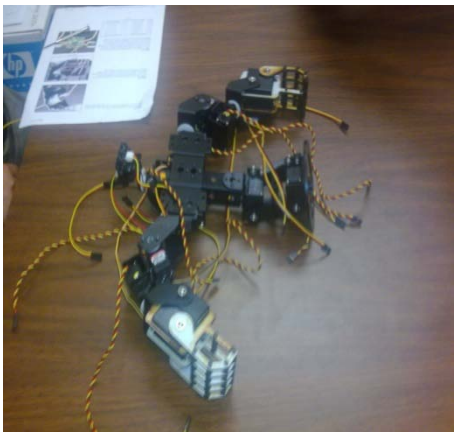
B-	Battery (-), black wire	6	Disable Timeout

4.3 **Mounting the Arm Base:**

We can now download and install Lynxterm. We plugged in the 6vdc 2amp wall pack and DB9 data cable.With Lynxterm installed, we can select channel 0 and move the slider to rotate the base. Before moving on, we need to press the "All=1500" button to re-center the base servo.

4.4 **Assembling the arm and hand for the robot:**

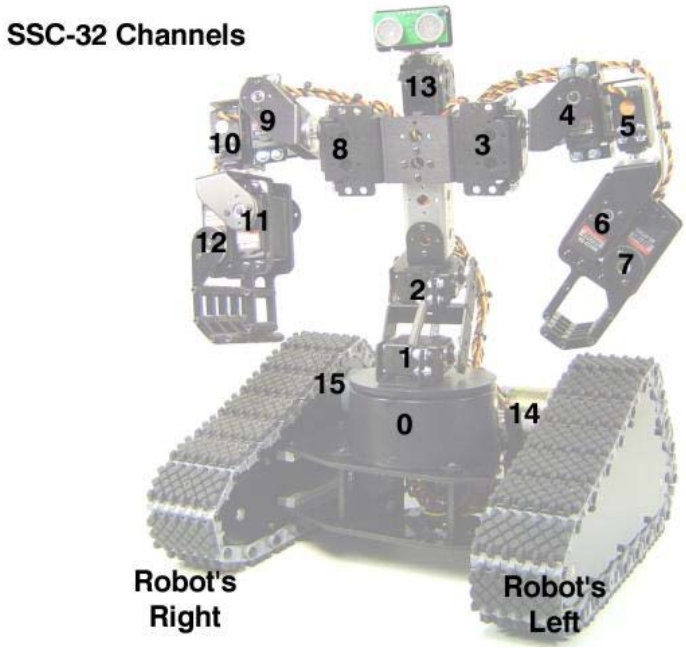
.



For the Johnny 5 kit, we used HS-645 servos, HS-475 servos in the position above the “thumb”, and H5-422 servos for the “fingers”.

4.5 **Assembling the Johnny 5 Torso:**

We attached the arms to the shoulder servos using four #2 x .250" tapping screws. On the robot's left side, we plugged the shoulder (up/down) servo into Channel 4 on the SSC-32, the forearm (rotate) into Channel 5, the wrist into Channel 6, and the hand (open/close) into Channel 7. On the robot's right side, we plugged the shoulder (up/down) servo into Channel 9 on the SSC-32, the forearm (rotate) into Channel 10, the wrist into Channel 11. We routed the servo wires through the aluminum channel.



4.7. Challenges faced during the project:

There were few challenges that we faced during the implementation and testing of the project, such as the following:

- **Installing the bot board II:** After successfully assembling the Johnny 5 robot, we realized that we needed a PS2 controller replacement. The Bot Board II required in order controlling the servo movements on the robot. The board was borrowed from the earlier Hexapod robot project. After connecting the Bot Board II to the SSC 32 processor chip, we started receiving data on the Atom IDE and we were able to control the servo movements on Johnny 5. Bot Board II has the capacity to handle up to 20 servos at the same time. We used 13 servos in our project.
- **The Sharp IR sensors functionality:** The multi-meter displayed voltage fluctuation when an obstacle was placed in front of the sensor, that is, the way it was supposed to. For doing the analog to digital conversion, our solution was to connect the sensors to the atom board in such a manner that it output digital data to the terminal. Values of 0, 127 or 255 were obtained. Where 0 indicated no obstacle in path, 127 was an obstacle in front of one of the sensors and 255 was the value for both sensors.
- **Multiple wires:** In the testing phase, we needed to take care of the multiple wires connecting the servos in order to have a full range of movements. Sometimes, some of the wires got interrupted due to the sudden moves of Johnny 5.
- **Tracks alignment:** It was difficult to keep the tracks of the robot aligned while moving. Potential reasons might be the frail body assembly or the non-perfect surface in our

laboratory. The result was that the Johnny 5 did not move in a straight path in most of the performed tests.

- **Charging batteries:** We needed to charge the batteries quite often in order to do extensive tests. The batteries kept discharging after every 3-4 trails of running the robot.

Despite the above enumerated challenges, our team enjoyed working to implement and test the Johnny 5 robot.

5. Experimental Results

Few experimental and testing results of Johnny 5 are provided in the video clips below.

movie.mp4

6. Basic Program

As mentioned in the Abstract, we called this application "Paparazzi". The mission of Johnny 5 is to hold a recording camera in his hands, go around the corner avoiding potential obstacles, take a picture of the objective, then come back to his original position. We list below the software loaded in his programmable memory for accomplishing this mission.

```
bsrt var word     ; base rotate                 PORT 0
lwbk var word     ; lower back                  PORT 1
upbk var word     ; upper back                  PORT 2
neck var word     ; neck                        PORT 3
rtsh var word     ; right shoulder        PORT 4
rtue var word     ; right upper elbow           PORT 5
rtle var word     ; right lower elbow           PORT 6
rtfg var word     ; right fingers               PORT 7
rttb var word     ; right thumb                 PORT 8
ltsh var word     ; left shoulder               PORT 9
ltue var word     ; left upper elbow            PORT 10
ltle var word     ; left lower elbow            PORT 11
ltfg var word     ; left fingers                PORT 12
lttb var word     ; left thumb                  PORT 13
rttk var word     ; right track                 PORT 14
lttk var word     ; left track                  PORT 15
left var word     ; to check the direction where to move the next movement
right var word    ; 'right'
cnt   var word    ; count of to and fro journey
init var word     ; initilization value, default set to 1500
movetime var word

bsrt = 1500
lwbk = 1550
upbk = 1400
neck = 1500
rtsh = 1400
rtue = 2000
rtle = 2300
rtfg = 1900
rttb = 1200
```

```
ltsh = 1400
ltue = 2000
ltle = 2500
ltfg = 1000
lttb = 1500
rttk = 1450
lttk = 1450
cnt  = 0
init = 0

movetime = 10000
serout p15,i38400,["#0P",DEC bsrt,"T",DEC movetime,13]
serout p15,i38400,["#1P",DEC lwbk,"T",DEC movetime,13]
serout p15,i38400,["#2P",DEC upbk,"T",DEC movetime,13]
serout p15,i38400,["#3P",DEC neck,"T",DEC movetime,13]
serout p15,i38400,["#4P",DEC rtsh,"T",DEC movetime,13]
serout p15,i38400,["#5P",DEC rtue,"T",DEC movetime,13]
serout p15,i38400,["#6P",DEC rtle,"T",DEC movetime,13]
serout p15,i38400,["#7P",DEC rtfg,"T",DEC movetime,13]
serout p15,i38400,["#8P",DEC rttb,"T",DEC movetime,13]
serout p15,i38400,["#9P",DEC ltsh,"T",DEC movetime,13]
serout p15,i38400,["#10P",DEC ltue,"T",DEC movetime,13]
serout p15,i38400,["#11P",DEC ltle,"T",DEC movetime,13]
serout p15,i38400,["#12P",DEC ltfg,"T",DEC movetime,13]
serout p15,i38400,["#13P",DEC lttb,"T",DEC movetime,13]
serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]

pause 2000

rtsh = 1900
ltsh = 1100
rttb = 1500
lttb = 1400
rtfg = 1300
ltfg = 1500
serout p15,i38400, ["#4P" , DEC rtsh,"T",Dec movetime,13]
serout p15,i38400, ["#9P" , DEC ltsh,"T",Dec movetime,13]
serout p15,i38400, ["#8P" , DEC rttb,"T",DEC movetime,13]
serout p15,i38400, ["#13P", DEC lttb,"T",DEC movetime,13]
serout p15,i38400, ["#12P", DEC ltfg,"T",DEC movetime,13]
serout p15,i38400, ["#7P" , DEC rtfg,"T",DEC movetime,13]

pause 2000

rtue = 2500
ltue = 2500

serout p15,i38400, ["#5P" , DEC rtue,"T",DEC movetime,13]
serout p15,i38400, ["#10P", DEC ltue,"T",DEC movetime,13]

pause 10000

rtfg = 1900
ltfg = 1000

serout p15,i38400, ["#4P" , DEC rtsh,"T",Dec movetime,13]
```

```
serout p15,i38400, ["#9P" , DEC ltsh,"T",Dec movetime,13]
serout p15,i38400, ["#8P" , DEC rttb,"T",DEC movetime,13]
serout p15,i38400, ["#13P", DEC lttb,"T",DEC movetime,13]
serout p15,i38400, ["#12P", DEC ltfg,"T",DEC movetime,13]
serout p15,i38400, ["#7P" , DEC rtfg,"T",DEC movetime,13]

pause 10000

loop:
     rttk = 1200
     lttk = 1200
     movetime = 50
     my_input var Word
     my_input = 0
     serout s_out, i9600, [DEC my_input,13]
     serout p15,i38400,["#2P",DEC upbk,"T",DEC movetime,13]
     serout p15,i38400,["#1P",DEC lwbk,"T",DEC movetime,13]
     owin P0,0,exit,[my_input]

    if ( my_input > 0 ) then
      rttk = 1450
      lttk = 1450
      serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
      serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
      neck = 700
      serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
      pause 2000
      owin P0,0,exit,[my_input]
      if ( my_input > 0 ) then
            neck = 2300
            serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
            pause 2000
            owin P0,0,exit,[my_input]
            if ( my_input > 0 ) then
                  neck = 1500
                  serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
                  rttk = 1450
                  lttk = 1450
                  serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
                  serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
                  goto exit
            else
                  rttk = 1800
                  lttk = 1200
                  serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
                  serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
                  pause 1600
                  neck = 1500
                  rttk = 1450
                  lttk = 1450
                  serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
                  serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
                  serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
                  pause 2000
                  goto loop
            endif
      else
```

```
            rttk = 1200
            lttk = 1800
            serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
            serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
            pause 1600
            neck = 1500
            rttk = 1450
            lttk = 1450
            serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
            serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
            serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
            pause 2000
            goto loop
        endif
    else
        rttk = 1100
        lttk = 1050
        serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
        serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
    endif
      serout s_out, i9600, [DEC my_input,13]
      pause 200

goto loop

 exit:
 cnt = cnt + 1;
 if ( cnt < 2 )   then
  pause 3000
  rttk = 1200
  lttk = 1780
  serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
  serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
  pause 3400
  neck = 1500
  rttk = 1450
  lttk = 1450
  serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
  serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
  serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
  pause 2000
  goto loop
 else
  pause 4000
  neck = 1500
  rttk = 1450
  lttk = 1450
  serout p15,i38400,["#15P",DEC rttk,"T",DEC movetime,13]
  serout p15,i38400,["#14P",DEC lttk,"T",DEC movetime,13]
  serout p15,i38400,["#3P" ,DEC neck,"T",DEC movetime,13]
  pause 2000
  movetime = 1000
  rtfg = 1450
  ltfg = 1450
  serout p15,i38400, ["#12P", DEC ltfg,"T",DEC movetime,13]
  serout p15,i38400, ["#7P" , DEC rtfg,"T",DEC movetime,13]
  clap:
```

```
  pause 1000
  rtue = 2500
  ltue = 2500
  serout p15,i38400, ["#5P" , DEC rtue,"T",DEC movetime,13]
  serout p15,i38400, ["#10P", DEC ltue,"T",DEC movetime,13]
  pause 1000
  rtfg  = 1900
  ltfg = 1000
  serout p15,i38400, ["#12P", DEC ltfg,"T",DEC movetime,13]
  serout p15,i38400, ["#7P" , DEC rtfg,"T",DEC movetime,13]
  pause 1000
  rtue = 2000
  ltue = 2000
  serout p15,i38400, ["#5P" , DEC rtue,"T",DEC movetime,13]
  serout p15,i38400, ["#10P", DEC ltue,"T",DEC movetime,13]
  goto clap
endif
```

7. Conclusion and Future Works

In this project, we assembled and programmed Johnny 5 in order to detect obstacles, take left, right and 180 degree turns, depending on the direction in which it finds the obstacle.

In the future, we would like to add more sensors such as the line sensor to make the robot walk exactly in the line and path specified. We would also like to improve from GP2D12F, to a better sensor and to move Johnny in fewer angles than 90, so that it can just walk past the obstacle.

8. Acknowledgments

We thank Mr. Jim Frye from Lynxmotion, Inc., for his permission to use text and images from his website ([2]) to write this technical report. In addition, we appreciate his valuable support and comments provided by email and phone discussions.

9. References:

1. Stefan Andrei: Lectures notes for 'Embedded Systems' Class (COSC-4301-01/COSC-5340-01), Summer of 2010: Embedded Systems. *Lamar University*, Department of Computer Science, Beaumont, Texas
2. ***, [on-line: http://www.lynxmotion.com/]
3. ***, [on-line: Johnny 5 Robot site: http://www.lynxmotion.com/category.aspx?categoryid=103, http://www.lynxmotion.net/viewforum.php?f=40&sid=4b74c41df8d5ec7b887e993836da4448
4. Basic Micro Chip - http://www.basicmicro.com/