Humanoid Android Robot Subsystems

Wenbo Wang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-128
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-128.html

May 22, 2014

Acknowledgement

Don Wroblewski
Ariel Bentolila

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2014

Electrical Engineering and Computer Sciences

Humanoid Android Robot Subsystems

Wenbo Wang

This Masters Project Paper fulfills the Master of Engineering degree requirement

Approved by:

1. Capstone Project Advisor:

Signature: ______________________________ Date ____________

Print Name: Donald Wroblewski
Department: Fung Institute

2. Faculty Committee Member #2:

Signature: ______________________________ Date ____________

Print Name: Ruzena Bajcsy
Department: Electrical Engineering and Computer Sciences

Abstract

There is a growing need for robots in many different sectors of industry. As demand increases and technology improves there will be a great demand for robots that can better integrate into the workplace and the home. Humanoid robotics are a potential technology that can bridge this gap. Our sponsor, Bay Area IP LLC, is exploring new Intellectual Property potential in this exciting field. We have designed and prototyped many humanoid robot components that can be used as a platform for exploring potential technologies as wells as serve as sources of new IP. This report describes the prototype robot leg and embedded software system for the robot platform, as well as summarizing the other subsystems worked on by the team.

Contents

1 Introduction

Robots are being widely adopted by in many industries, and there is a growing need for humanoid robotics. Humanoid robotic components can have many applications such as manufacturing and prosthetics. A key component of our project, the mechanical design of a humanoid robotic hand can be applied to both manufacturing and prosthetics. Currently humanoid robots and robot subsystems are not in wide use, so it is a good target for our sponsor, Bay Area IP, which is looking do develop intellectual property that they could license to others.

There are many technical challenges that come with trying to reproduce the human form. The human body is very complex and there are many different components that need to interact and function correctly. The human hand has 27 degrees of freedom, and it is very difficult to reproduce that complexity [1]. However, modern manufacturing techniques such as 3D printing can get us closer to reproducing that complexity. Reproducing human capabilities through software is also a great challenges. It is difficult to design and implement complex control systems capable of mimicking the human body. New computer hardware is allowing us to create more complex robots capable of performing human like tasks.

Our sponsor, Bay Area IP, envisions three main components to the robot: a computer vision system, a complex humanoid robotic hand, and a high performance bipedal locomotion system. The hand would be a light weight and have a high degree of freedom. The computer vision system will recognize objects in real time and guide the robotic arm towards the object and guide the hand for grasping. The robotic legs will be able to perform complex gaits and high speed motions such as running. The role of our team is to lay the foundations for these three systems, and try to do as much as we can do complete these three subsystems.

The subsystems we designed were the arms, hands, legs, feet, and vision systems. We designed the mechanical structure of these components and the software for controlling the different parts. For the hands we did CAD design of the hand structure. We also performed tests on using shape memory alloy(SMA) to actuate components of in the fingers. Simple prototypes of the arms and legs were constructed using off-the-shelf components to provide testing platforms for the software. The control software was simulated in MATLAB and then implemented in C++ on a microcontroller. The computer vision system was implemented on an AMD APU, and utilized a novel laser system to augment the computer vision algorithms.

To control our robot, we utilized a powerful 32 bit microprocessor, the PIC32MX795F512L which allows us to perform complex calculations in real time [2]. This allows the robot to perform complex computer vision tasks and control algorithms. With the PIC32 as the foundation I built the firmware for the robot, implementing the fundamental sensory and motor controls of the humanoid robot. The first step was to implement the necessary communication protocols to connect the PIC32 with the other software components: I2C to integrate other integrated circuit chips, USB to communicate with the PC, and UART to communicate with other controllers. These systems cover nearly the full spectrum of communication protocols used by common ICs and components used for embedded software and robotics applications, and they allow us to easily integrate new components and features into the robot hardware.

The motor functions of the robot are mainly governed by a collection of servos on the arms and legs. To control the servos, I utilized the SSC-32 servo controller, a powerful controller capable of synchronizing the motions of 32 servos at the same time [3]. A single SSC-32, controlled through a serial UART connection, would be able to control the motions of the arm and legs and synchronize them with millisecond precision. The sensory information consists of accelerometer and gyro readings from the MPU6050. The MPU6050 is connected through an I2C connection, and it consists of a 3-axis gyro and a 3-axis accelerometer, as well as temperature sensors. The MPU6050 can also be connected to other sensors such as magnetometers [4]. The MPU6050 can be placed on the arms and legs to generate sensory feedback for controlling the motions of the legs and arms.

The team completed the bipedal legs of the robot, and I was able to test the controller on the bipedal system. The control of walking was based on a linear combination of different sine waves which governed the periodic motion of the legs. Through system testing, we discovered that the robot has very high power requirements, and the power supplies we had could not power the entire system. Much more powerful power supplies will be needed to complete full system tests. The report describes the detailed design of the firmware and the robotic legs on which the firmware was tested, as well as the challenges we faced and our recommendations to the sponsor for moving forward with the project.

2 Literature Review

2.1 Robot Software Frameworks

The software for the robotics industry is mostly made of code custom tailored to specific hardware and specific industries, unlike traditional PC software which tend to be multi-platform and independent of the underlying hardware. Some reasons for this are the relative youth of this industry and the very specific niches that robots manufacturers fill. Many robots are designed for specific manufacturing and assembly purposes, where the customers are large corporations willing to pay high prices for these robots. Corporations are usually seeking reliability in their manufacturing robots, so custom tailoring the code for specific hardware has much greater benefits than sacrificing performance for portability. Comparatively, hobbyists and researchers seeking multi-platform and easy to use software is a much smaller segment [5].

As robots are becoming more complex and difficult to program, many individuals have tried to create software frameworks to ease the process [6]. Easy to use software can be a main selling point for some robots; the Baxter robot from Rethink Robotics lists ease of programming as one of it's key features [7]. One framework is the ROS which is a Robotic Operating System, which implements some of the features you would find in a traditional operating system in order to facilitate development of robot software. The ROS is open source, free, and language independent, meaning that it can be used with many different programming languages. The ROS is structured around libraries and drivers like a normal OS, so the programmer and write their code to be portable to many different hardware systems, like a software for a PC [6].

Another framework is the Miro developed by Utz et al., which uses a object oriented paradigm. The Miro, like the ROS, is open source. Miro implements all the basic fundamentals of object oriented software, such as information hiding, abstractions, polymorphism, and inheritance. The Miro provides a familiar framework that hides the details of the underlying hardware from the programmer [8].

The framework we will be using is the Multi-Platform Integrated Development Environment (MPIDE). This framework is a microprocessor programming framework designed to work for many different microprocessors. The MPIDE is based on the Arduino framework, which is based on C++. This is a very popular framework for programming microcontrollers since it offers a simple API abstraction for the complex microprocessor hardware. Being based on C++, the MPIDE offers a object oriented programming pattern like the Miro. The Arduino framework, on which MPIDE is based, has a large community of enthusiasts offering a strong base of support and software libraries. The MPIDE has the ease of use of the Arduino while also supporting a larger variety of much faster microprocessors [9].

2.2 Computer Vision

Object detection has been well studied in a variety of applications. Many machine learning algorithms are utilized for object detection such as Support Vector Machines or Convolutional Neural Networks[10],[11]. Many of the object detection algorithms utilize Histogram of Oriented Gradient(HOG) features, which are based on taking a gradient over the image and generating histograms of gradient angles and magnitudes using overlapping sections of the image[12].

Complex object detection algorithms can be very resource intensive, as they often need to scan the image multiple times at different resolutions to find all the objects[13]. This can be a very expensive process on an embedded system such as the robot. In a cluttered space it can be very difficult to detect objects correctly in real time. On a robotic system there usually exist other sensors that can supplement the visual information to help ease this process. Utilization of depth sensors can help process the visual information more easily[14].

Computer vision techniques could have many applications for manufacturing robots. As manufacturing robotics advance, they will need more accuracy and intelligence to accomplish their tasks, and vision is a good tool for accomplishing this. Already there are manufacturing robots utilizing vision for tasks such as part recognition and sorting[15].

2.3 Robotic Hand

There are many different companies and research groups constructing robotic hands for many different applications. There are hands designed to be prosthetics, as well as hands designed to be integrated into robotic systems. One of the most complex is the UB Hand IV, this hand is nearly the same as a human hand in terms of capabilities. However, the hand is very heavy and very power hungry, so it is not very suitable for most applications[16].

However, most light weight hands are lacking in complexity and function. The this recently developed lightweight hand only has five degrees of freedom[17]. To develop a hand that is both light weight, small, and fully functional will require new design techniques.

2.4 Shape Memory Alloy

We want to leverage some modern technology to make a robotic hand that is smaller and light weight. One potential tool is Shape Memory Alloy (SMA), a type of material that can expand or contract when heated[18]. Many metallic alloys exhibit the shape memory effect[19]. These alloys can be deformed at a lower temperature, when heated, they will return to their original shape. We can use this do accomplish actuation using a small form device heated through electrical current[20]. This actuation can be used in the fingers of the hand , where space is very limited.

2.5 Robot Leg

A popular control method for robot locomotion is Central Pattern Generators (CPG). Central pattern generators are neural networks that generate periodic control signals in the body. This control can be generated in the absence of feedback signals. The human body utilizes this system for many different rhythmic motor functions that the body needs to perform[21].

Central pattern generator can also be adapted to the locomotion of robots. CPG can be used to control locomotion of bipedal robots, as well as hexapods and octopods. Using coupled oscillators one can design a control system for bipedal robots. CPG are well suited for feedback control of bipedal locomotion. Properly implemented CPG also allow for higher level control of the walking without needing to worry about exact servo outputs. However CPGs are not well understood and difficult to design properly[22].

2.6 Robot Arm

For the controls of our robot arm we will be implementing Fuzzy Logic control systems. These control systems can handle nonlinearities well, and they are well studied for control of robotic systems such as arms[23]. Fuzzy logic controllers can also be used to control balance of the robotic legs, treating the legs as an inverted pendulum[24].

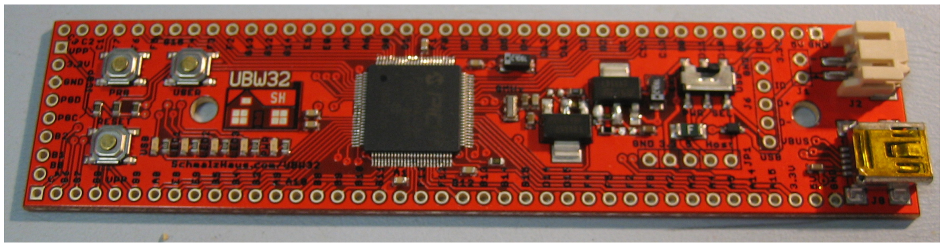


Figure 1: UBW32 board from SparkFun electronics[26]

Rather than using complex models, a fuzzy logic controller relies on empirical rules, this makes the controller computationally cheap and well suited for embedded applications. A fuzzy logic controller has three main components: a fuzzifier, a rule base, and an defuzzifier. In the fuzzifier, analog inputs are fuzzified into fuzzy logic values between 0 and 1. The fuzzy values are put into the rule base to determine a set of fuzzy outputs, and then the defuzzifier combines the outputs into an analog output[25].

3 Materials and Methods

3.1 Materials

PIC32MX795F512L A 32 bit micro-controller for performing the control algorithms of the robot. The PIC32MX795F512L comes from SparkFun electronics as part of the UBW32 board, which comes preloaded with a bootloader for programming the board. See figure 1 for an image of the UBW32 board. The PIC32MX795F512L is programmed in C/C++. To make the device easier to use, we loaded an avrdude bootloader onto the device to use the MPIDE software. The MPIDE framework allows us to use Arduino libraries on the PIC32MX795F512L which helps to accelerate development of basic I/O software for the board[2].

PICKIT3 A hardware programmer for the PIC32MX795F512L microprocessor. See figure 2. The pickit3 and program and debug the PIC32 micro-controller. Used to download code onto the UBW32. Useful for installing new bootloaders onto the UBW32 and for restoring broken firmware[27].

MPU6050 A gyro and accelerometer for obtaining information about the kinematics of the arm and the leg. See figure 3. Uses I2C connection to the PIC32MX795F512L for

Figure 2: PICkit 3 In-Circuit Debugger from Microchip[28]

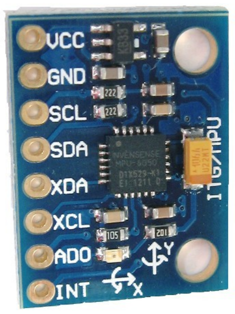


Figure 3: MPU-6050 Triple axis accelerometer and gyro[29]

communication[4].

HSR-5498SG Servos from Hitec. Utilized to control arm and leg joints. See figure 4. The servos require 6-7 Volts. Each servo requires at least 200mA when running without load, and over 1A when stalled. The actuation of the arm is accomplished by mini-motors rather than servos[3].

Figure 4: HSR-5498SG servo from Hitech[**hsrfigcite**]

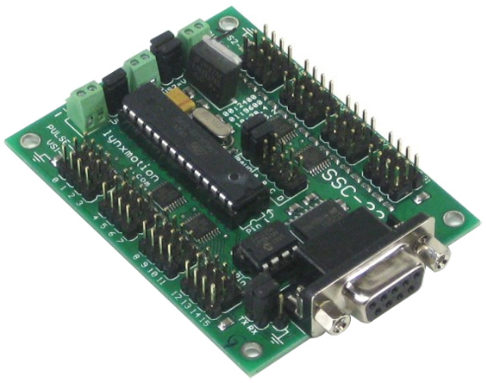


Figure 5: SSC-32 servo controller from Lynxmotion[31]

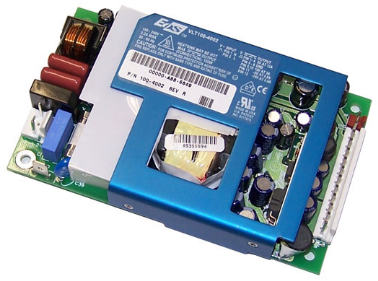


Figure 6: VLT100-4002 power supply[33]

SSC-32 Servo controller from Lynx motion. The on-board controller is an Atmega168-20PU. See figure 5. The SSC-32 is controlled using serial signals from another microprocessor or the PC. The control signal is a string containing a series of commands. The controller can specify pin number, servo position, rotation speed, and rotation time. It can power 32 servos simultaneously[30].

VLT100-4002 Power supply for powering the servos. See figure 6. 5V output capable of supplying 3A to 12A of current[32].

Yihua 1502DD Adjustable power supply for servos. See figure 7. 0-15V output and 0-2A output[34].

YH-1502DD 15V/2A DC POWER SUPPLY
YIHUA
DC POWER SUPPLY PS-1502DD
Factory Audit
Alibaba Production Audit report
ISO9001-2008
CE Certificate

Figure 7: Yihua 1502DD DC power supply[35]

3.2 Methods

3.2.1 Microcontroller Setup

To program the PIC32MX795F512L, we used the Multi-Platform Integrated Development Environment(MPIDE) from chipKIT. To use MPIDE with the program, we had to program the PIC32 with the avrdude bootloader found here: `https://github.com/chipKIT32/PIC32-avrdude-bootloader`. Using the PICkit3, we loaded the new bootloader onto the UBW32 and was able to use MPIDE to program the board. MPIDE utilizes the Arduino libraries and Arduino style C++ code for programming the board. Through MPIDE, there are a wide variety of libraries available for performing basic I/O tasks using the PIC32.

3.2.2 MPU-6050 Setup

The MPU-6050 utilizes an I2C connection to the read and write data. The device readings are stored on a 1024 byte FIFO buffer on the MPU[4]. The MPU can be used to acquire sensory information for controlling the motions of the legs and the arms. We set up the MPU6050 using the PIC32 I2C libraries in MPIDE. We utilized the `Wire.h` library to read and write to the I2C bus of the PIC32[29]. There are a total of five SDA pins on the PIC32MX795F512L, so we can have a minimum of five of these devices connected to our microcontroller.

3.2.3 Servo Setup

The servos each had a range of 180 degrees. The SSC-32 represents this as a value between 500 and 2500. Due to the mechanical construction most joints had a limited range of motion. Most joints had a range of 100 degrees. The exact ranges were tested and calibrated after the completion other robot leg. Each servo requires approximately 6V and 200mA to 1A of current depending on the load[30]. For our tests, we used the Yihua DC power supply which is adjustable up to 15V and supplied up to 2A[34].

3.2.4 Servo Control

To control the servos we utilized a SSC-32 servo controller. The controller has 32 channels for servo control. Each channel can be given a position, rotation speed, and rotation time. Multiple servo channels can be controlled simultaneously through a single command The SSC-32 is powered through a 9V battery. The controls are given in the form of strings, the following is an example of the commands used for SSC-32.

```
#5 P1600 T1000 <cr>
```

The #5 defines the channel number which ranges from 0 to 31. The P1600 sets the position of the servo from 500 to 2500. The T1000 sets the time of rotation in milliseconds. The carriage return(`<cr>`) signifies the completion of a command. The following is an example command for simultaneous movement of servos:

```
#5 P1600 #10 P750 T2500 <cr>
```

In this case all the servos adjust their speed so they arrive at the specified position in the specified time.

3.2.5 Walking Algorithm Implementation

The walking algorithms were designed by Zhu Ziqi, a fellow team member. He designed and simulated the algorithms in MATALB and SIMULINK and I implemented the algorithms in C++ on the PIC32MX795F512L. The servo positions were determined from a linear combination of sine waves. For our preliminary walking algorithms we only considered the knee and hip joints. The SSC-32 position control signal requires a integer ranged from 500 to 2500, corresponding to 180° of motion[3]. Due to the mechanical construction of the arm and leg joints, which mimicked human structure, most joint servos were limited to a range between 1250 and 2500. The output from the sine waves would be converted to a range in the SSC-32's output and then sent to the servos. Equations 3 and 4 show the equations for calculating the position of each servo as a function of time. The angle of the hip is a degree relative to the vertical axis of the 3D space, while the angle of the knee is relative to the axis along the thigh of the robot.

$$\begin{aligned}
\phi_3(t) &= a_{13L}sin(b_{13L}t + c_{13L}) + a_{23L}sin(b_{23L}t + c_{23L}) + a_{33L}sin(b_{33L}t + c_{33L}) \\
&+ a_{43L}sin(b_{43L}t + c_{43L}) + a_{53L}sin(b_{53L}t + c_{53L}) + a_{63L}sin(b_{63L}t + c_{63L}) \\
&+ a_{73L}sin(b_{73L}t + c_{73L}) + a_{83L}sin(b_{83L}t + c_{83L}) \qquad (1) \\
\phi_4(t) &= a_{14L}sin(b_{14L}t + c_{14L}) + a_{24L}sin(b_{24L}t + c_{24L}) + a_{34L}sin(b_{34L}t + c_{34L}) \\
&+ a_{44L}sin(b_{44L}t + c_{44L}) + a_{54L}sin(b_{54L}t + c_{54L}) + a_{64L}sin(b_{64L}t + c_{64L}) \\
&+ a_{74L}sin(b_{74L}t + c_{74L}) \qquad (2) \\
\theta_{hip}(t) &= \frac{\phi_3(t)}{180\pi} \qquad (3) \\
\theta_{knee}(t) &= \frac{\theta_{hip}(t) - \phi_4(t)}{180\pi} \qquad (4) \\
& \qquad (5)
\end{aligned}$$

The conversion from angle in degrees to servo position is given by the following equation.

$$P = \frac{\theta}{90} * 1000 + 1500 \qquad (6)$$

The positions of the legs are calculated by the PIC32 and sent to the SSC-32 through the Universal Asynchronous Receiver Transmitter(UART) module of the PIC32. The UART is a serial connection between the PIC32 and the SSC-32. The calculated results are composed into a string in the form of an SSC-32 command and sent to the servo controller.

3.2.6 Testing

Tests were done on the control system and servos to make sure every system function correctly. Software performance was tested using the hardware timers on the PIC32. The performance of individual processes were timed to identify performance bottlenecks and possibilities for optimization. The main processes we timed were the calculation of the sine waves for robot walking and serial transmission between devices. We also measured overall runtime for each controller iteration to make sure physical motion of the robot was synchronized with the controller update time.

Servos were also tested for performance. While testing the basic servo functionality of the controller, I used a small battery to power the servo, as their was little load to lift. Initially I tested to check that the P500, P1500, and P2500 settings of the SSC32 matched the 0, 90, and 180 degrees of the physical servo. Initial servo testing was done by sending manual commands to the SSC32 controller through a serial terminal.

After the legs were assembled, I tested the servos on the legs joints. These tests were performed using the Yihua 1502 power supply which could support the load of lifting the legs. For these tests I also sent direct commands through a serial terminal to control the positions of the individual joints. I tested the maximum and minimum angles of each joint servo. This was done by iteratively trying the different possible joint angles until we found the all positions that would not be constrained by the mechanics of the joint assembly. Once I found a position value that the servo could not move to due to mechanical constraints, we tested narrowed the search to find the maximum allowable position. We also tested to find all the neutral positions of the servo, where neutral meant standing upright. These maximum and minimum position values, would be coded into the controller as software limits, as trying to push the servo against these limits wold likely overload and damage the servo.

Overall system testing was done by attaching all joint servos to the SSC32 and driving them with the VLT100 which could supply up to 20A of current. Initial testing was done using serial terminal was manual control. First we test the ability to synchronize the motions of multiple servos at once using the serial terminal to send synchronized commands. Then we switched to the autonomous mode and tested the controller's ability to correctly generate periodic motions in the legs. To make sure the PIC32 was outputting the correct commands to the SSC32, we had the PIC32 mirror those commands to the PC, where we can check the output position and command syntax through a serial terminal.

4 Discussion

4.1 Overall System

We have components of a robotic platform prototyped and designed. The Robotic legs and arm are prototyped and constructed. The hand and feet have been designed in CAD, and The basic software and control framework has been established. The basic I/O and control tasks have been implemented in C++ on the microcontroller with easy to use APIs that others can use to control the completed robot legs. Overall we are at a good position for the sponsor to take the foundations we have built and try to implement different control algorithms and machine learning algorithms.

4.2 Robot Leg

The leg was constructed using with six servos on each leg. Three servos on the hip mimicked the ball and socket configuration of the human hip joint, which has three degrees of freedom[36]. The three modes of motion are along the sagittal, frontal, and transverse planes[37]. The knee has one degree of freedom, and the ankle has two degrees of freedom in the sagittal and frontal planes. Each leg has six degrees of freedom in total, this is fine for most gaits on uneven ground, but more complex terrain may be difficult to manage; More degrees of freedom may be needed to obtain more human like gait on difficult to traverse terrains[36].

The shins and pelvis were constructed from aluminum U channels. The thighs where made from aluminum cylinders, and the feet were aluminum plates (Figure 8). The SSC-32 servo controller was attached to the hip. A tether connects the servo controller to the power supply and the PIC32 controller. Figure 9 shows the configuration of the servos on the leg, as well as their direction and range of motion in terms of SSC-32 position values. The full range of valid position values was determined through testing each individual joint. I found that exceeding these values generally breaks the logic of the SSC32, and the controller must be reset as a result.

4.3 Firmware

The completed firmware components are the serial communications to different devices, servo control, and accelerometer/gyro control. The communications channels implemented are the USB communications between the PIC32 and the PC, the UART connection between the PIC32 and the SSC-32 servo controller, and the I2C connection to the MPU6050 sensors. The I2C connection can be adapted to more sensors that may be added in the future for other devices.

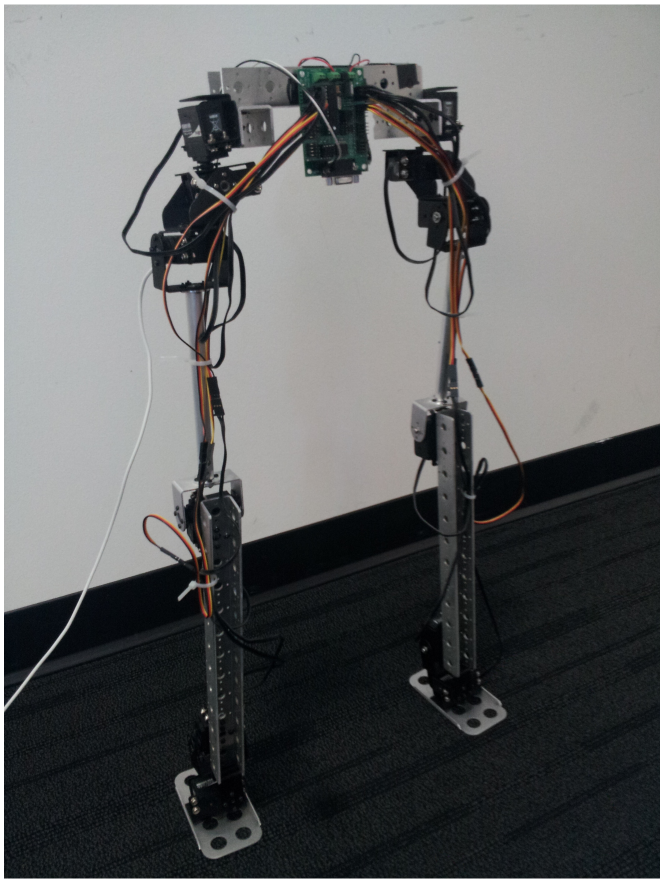
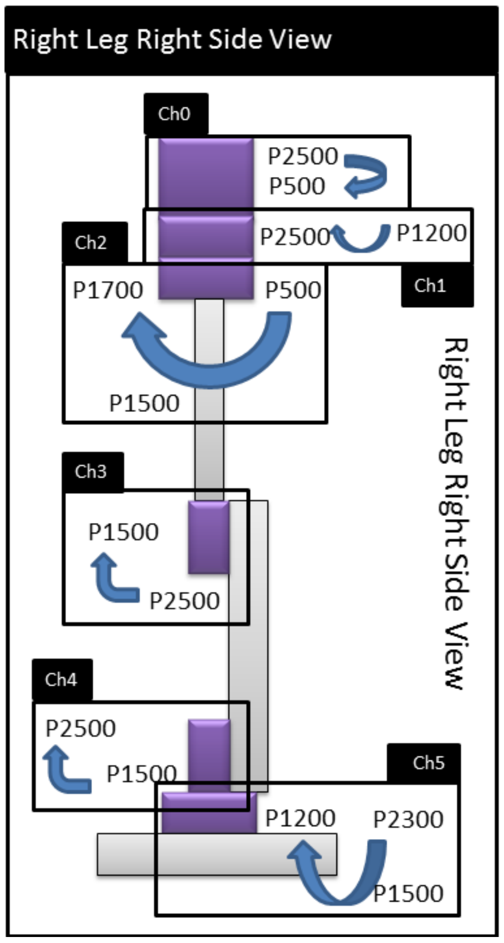


Figure 8: Prototype leg constructed of aluminum. Total of 6 degrees of freedom on each leg. The servo controller is attached to the hip



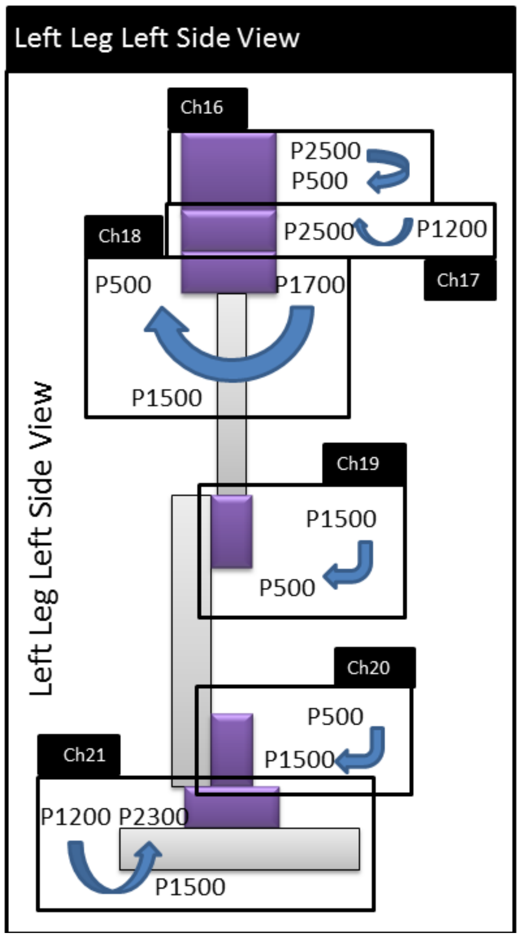


Figure 9: Location of the servos on the leg, and the directions and ranges of motion for each servo in terms of the SSC-32 position outputs

The servo controls are accessed through `SSCServo.h` which contains functions and settings for controlling servo position through the SSC-32 controller. The MPU6050 is accessed through `MPU6050.h` which defines memory addresses of the MPU6050 registers and functions for reading the output of the sensors and configuring the sensor settings. Users of the robot can adapt these settings to program and prototype their own control algorithms.

4.4 Locomotion Controller

Ziqi performed the simulations of the robot leg locomotion algorithms, and obtained the optimal parameters for equations 1 and 2. The following are the parameters we used. Refer to Ziqi's report for a detailed analysis of the control system.

```
phi_3:           phi_4:
a13L =   73.11;  a14L =  14.91;
b13L =   19.69;  b14L = 0.8727;
c13L = -0.6449;  c14L =  4.016;
a23L =   52.58;  a24L =  1.559;
b23L =  0.6307;  b24L =   19.4;
c23L =   2.671;  c24L = -1.197;
a33L =   12.35;  a34L =  27.52;
b33L =   12.05;  b34L =  15.42;
c33L =  -2.836;  c34L = 0.9818;
a43L =   3.357;  a44L =  28.88;
b43L =   29.81;  b44L =  45.43;
c43L =  -0.797;  c44L = -5.819;
a53L =   62.63;  a54L =  15.08;
b53L =   20.15;  b54L =  31.49;
c53L =   2.236;  c54L =  1.844;
a63L =   12.04;  a64L =  1.769;
b63L =   5.405;  b64L =  9.852;
c63L =   3.598;  c64L = -5.219;
a73L =   1.786;  a74L =   31.1;
b73L =   34.76;  b74L =  45.58;
c73L =   1.995;  c74L = -2.751;
a83L =   1.418;
```

```
b83L =    47.2;
c83L =  -0.402;
```

The robot is controlled through a serial connection to the computer. The PIC32MX795F512L is connected to the PC through the USB connection on UART channel 0. Commands can be passed into the robot from a serial terminal on the PC. There are three modes of operation: Standing, Walking, and Direct Control. Activated by sending the strings `stand`, `walk`, and `direct` through a serial terminal to the PIC32. While in stand mode, the robot holds a rigid upright position. While in walk mode, the robot attempts to walk according to the sine waves laid out by equations 1 to 4. In direct mode, each servo can be commanded directly using the SSC-32 commands; in this mode all the commands passed to the PIC32 are redirected to the SSC-32. Servos on the right leg are attached to channels 0-5 on the SSC-32 and the servos on the left leg are on channels 16-21. Figure 9 shows the corresponding channels for each servo. Figure 10 shows an overview of how the circuit components are connected in the legs.

4.5 System Analysis

The controller performs simple round robin scheduling when calculating servo outputs. It looks at each servo in order and calculates the correct output. The final command is composed from the servo outputs and sent as a single synchronized motion command to the SSC-32. The execution speed of the round robin loop is determined in the code. To get smooth motions in the joints, we needed to calibrate the timing of the commands sent to the SSC-32. We are utilizing a 100ms time step for the position calculations, and we found that approximately 94ms update time is sufficient for smooth motions. To smooth out the motions, the update time needs to be shorter than the motion time programmed into the SSC-32. We calculate the necessary position of the servos and construct the command and send the command to the SSC-32 in 94ms. We have 12 servos, and each servo requires around 8 bytes of data to be sent. The PIC32 runs at 80MHz, so it is sufficient for our current configuration, the calculation time negligible, we measured times in the range of a few milliseconds for the calculation of the sine functions of all the servos [2]. The serial transfer rate can be limiting since the SSC-32 requires a long string encoding the positions of all the servos that need to be moved, and each servo needs a string of approximately 8 characters, with 12 servos on the legs and 12 servos on the arms that is 1536 bits of information that needs to sent every time step. The SSC32 can be configured for different

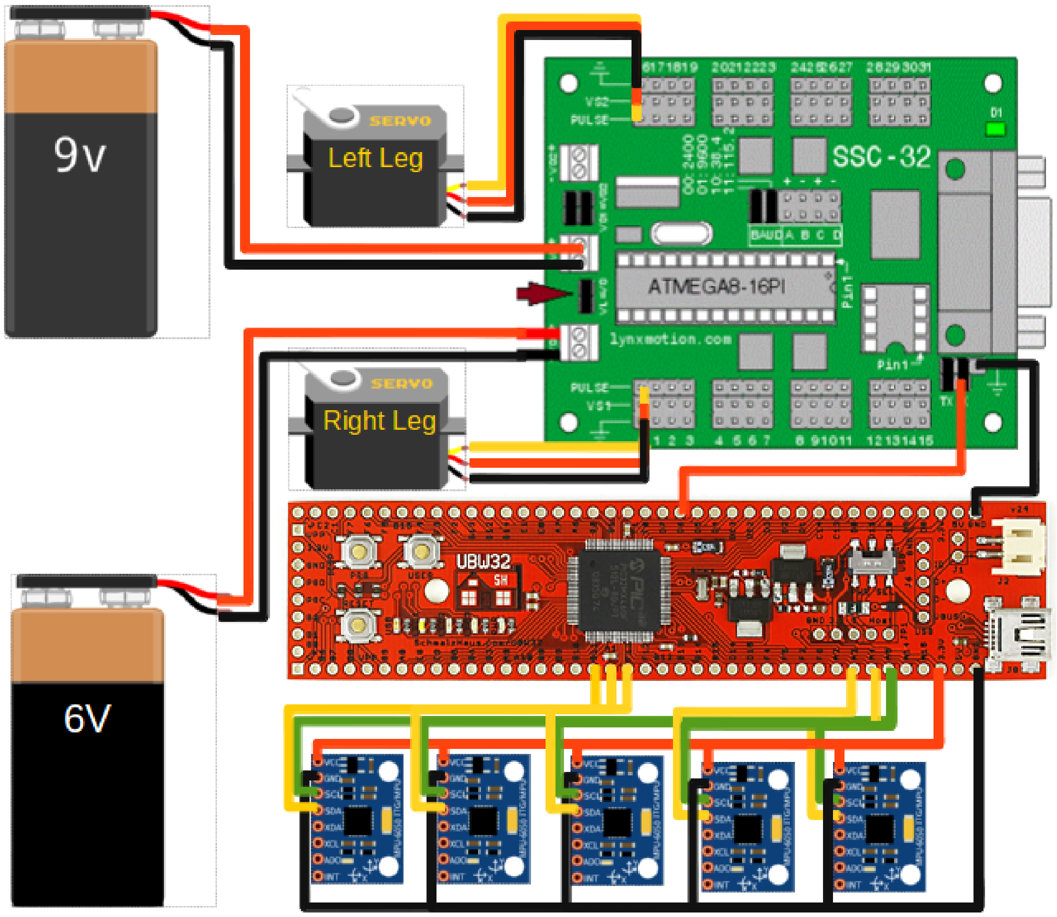


Figure 10: Overview of the circuit components and the connection

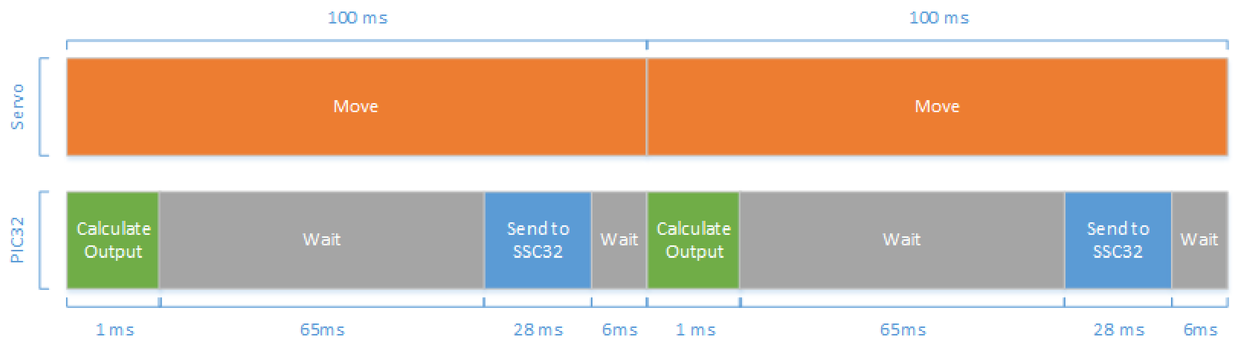


Figure 11: Timing of the processor steps with the motions of the servo

Baud rates of 9600, 38400, and 115200. Table 1 shows the measured transfer times at each of the available Baud rates. These values were measured using commands for simultaneous control of 12 servos.

Baud Rate (bps)	Transfer Time (ms)
9600	115
38400	28
115200	10

Table 1: The number of milliseconds to transmit a command for 12 servos to the SSC32 for different Baud rates

Figure 11 shows the final configuration of the firmware timing. We used 38.4kbps for the Baud rate in our prototype. The majority of the time is spent waiting, this time may be filled up with other tasks as the other components of the robot are completed.

The maximum update speed for the SSC-32 is 20ms, so that is the main performance limiter [38]. The 20ms limit should be fine for most motions, since 20ms is faster than most human knee motions, which requires nearly one second to move from full extension to full contraction [39]. However, having a faster update rate will increase the resolution of the motion and achieve a more human like gait. The SSC-32 is controlled by a Atmel ATMEGA168-20PU, which is a 8-bit 20Mhz microcontroller [40]. To increase performance, we may be able to shift the servo control to the PIC32, which is a much faster microcontroller, and we would not need the serial communications between the PIC32 and the SSC-32, but that would require more complicated task scheduling to synchronize the servo movements.

The robot's mobility is limited due to lack of proper power supply to the robot. We only have two power supplies, the Yihua 1502DD and the VLT100, neither of which is suitable for the task of powering the entire bipedal system. The Yihua can support full control of a single joint with great reliability. The datasheet of the servos list the stall current as 1.2A and the no load current as 200mA. Through testing we see that actuating a single knee servo to lift the leg requires over 1A of current. The Yihua has a maximum load of 2A, which cannot support motion of the robot. Testing with the VLT100 which is a 5V 3-12A power supply also proved useless. The power supply could not reliably lift the legs of the robot and hold their positions. This is likely due to the servos requiring 6V to 7.2V of supply voltage. The VLT100 needs 3A minimum load to sustain a stable voltage output, and even with 12 servos they do not draw 3A when the majority are unloaded. The leg has 12 servos in total, most of large motions will be in the four hip and knee joints, while the other joints will be responsible for smaller motions to balance the robot. The arms are of similar construction, but the elbow and shoulder joints would be responsible for a even larger load of lifting the arms in air.

Ideally we want to be able to supply 28.8A of current to satisfy the maximum requirement of the servos. We can opt for a high current power supply such as a BK Precision Model 1796, which can supply 0-16V and 0-50A which will be enough for our applications [41]. However such supplies are probably too bulky to be put on to the robot chassis, so the robot will need to be tethered to the table supply. We can also boost the voltage out put of the VLT100, the 12A output is probably sufficient for the controlling the legs or the arms, so we may need two supplies for the whole robot. The VLT100 is small enough to attach to the robot, but it needs to be attached to tethered to the wall supply. We can also use batteries to power the robot. The BP13-6 S from BB Battery Co. has 6V output and a 30A maximum discharge, and the battery is rechargeable [42]. The battery has the advantage of being mobile and can remove the necessity of tethering the robot, though the battery is fairly heavy at 5.5lb [42]. To fully free the robot from the tether, we will also need to replace the wired USB connection with wireless communication.

5 Conclusion

The robot has legs and arms that can be used as testing platforms for various algorithms. Future groups can utilize this robotic platform as a foundation for developing more complex control systems and artificial intelligence software. The overall vision laid out by the our industry adviser at the beginning of the project was of a humanoid robot that has advanced locomotion capabilities such as running and jumping and advanced vision and hand manipulation capabilities. The goal for our group was to build the foundations of this robot. We have successfully prototyped the simpler components like the leg and arm, and we have laid the software foundations for the control system and computer vision system. It would be easy for future teams to build on the software and hardware we have to complete the vision of our sponsor.

More testing and simulations need to be done to finish the control systems of the robot. More degrees of freedom can also be added to implement more human like motions. Once that is completed more work can be done to push it beyond the state of the art, by implementing complex motions such as bipedal running. The vision system needs to be completed and interfaced to the robot arm, so the vision can be used to guide the arm's motions. We faced difficulties in the mechanical design of the hand with the constraints of size and weight, especially in implementing SMA into the design, so more work and testing need to be done in this area.

The areas of advanced robotic locomotion, computer vision, and advanced humanoid hand and feet design are all great areas for the sponsor Bay Area IP to explore IP opportunities. These technologies can bring many benefits to areas such as manufacturing and prosthetics.

References

[1] George ElKoura and Karan Singh. "Handrix: animating the human hand". In: *Eurographics symposium on Computer animation* (2003). URL: http://dl.acm.org/citation.cfm?id=846291.

[2] *PIC32MX5XX/6XX/7XX Datasheet*. Microchip. 2012. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/61156H.pdf.

[3] *User Manual SSC-32 Ver 2.0*. Lynxmotion, Inc. 2005. URL: http://www.swarthmore.edu/NatSci/ceverba1/Class/e5/E5Lab2/ssc-32lynxmotionmanual.pdf.

[4] *MPU6050 Datasheet*. InvenSense Inc. 2013. URL: http://invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf.

[5] Choulsoo Jang et al. "OPRoS: A New Component-Based Robot Software Platform." In: *ETRI journal* 32.5 (2010).

[6] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. 2009.

[7] *Baxter*. Rethink Robotics, Inc. URL: http://rr-web.s3.amazonaws.com/assets/Baxter_datasheet_5.131.pdf.

[8] Hans Utz et al. "Miro-middleware for mobile robot applications". In: *Robotics and Automation, IEEE Transactions on* 18.4 (2002), pp. 493–497.

[9] Rick Anderson and Dan Cervo. "Using Arduino with PIC32 and ATtiny Atmel Chips". In: *Pro Arduino*. Springer, 2013, pp. 169–188.

[10] Tudor Barbu. "SVM-based human cell detection technique using histograms of oriented gradients". In: *cell* 4 (2012), p. 11.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *NIPS*. Vol. 1. 2. 2012, p. 4.

[12] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1467360.

[13] Pedro Felzenszwalb et al. "Visual object detection with deformable part models". In: *Communications of the ACM* 56.9 (2013), pp. 97–105. URL: http://dl.acm.org/citation.cfm?id=2494532.

[14] Stephen Gould et al. “Integrating visual and range data for robotic object detection”. In: *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*. 2008. URL: http://hal.archives-ouvertes.fr/inria-00326789/.

[15] *Factiories of the Future: Beyond the Limits of Industrial Robotics*. 2011. URL: http://www.cometproject.eu/publications/sir-industrial-robotics.pdf.

[16] C Melchiorri et al. “Development of the UB Hand IV: Overview of Design Solutions and Enabling Technologies”. In: *IEEE Robot Autom Mag* 20.3 (2013), pp. 72–81. DOI: 10.1109/MRA.2012.2225471.

[17] Takeshi Takaki and Toru Omata. “High-performance anthropomorphic robot hand with grasping-force-magnification mechanism”. In: *Mechatronics, IEEE/ASME Transactions on* 16.3 (2011), pp. 583–591.

[18] L McDonald Schetky. “Shape-Memory Alloys”. In: *Kirk-Othmer Encyclopedia of Chemical Technology* (1982). URL: http://onlinelibrary.wiley.com/doi/10.1002/0471238961.1908011619030805.a01/full.

[19] CM Wayman. “Shape memory alloys”. In: *MRS bulletin* 18.04 (1993), pp. 49–56. URL: http://journals.cambridge.org/abstract_S0883769400037350.

[20] Koji Ikuta. “Micro/miniature shape memory alloy actuator”. In: *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE. 1990, pp. 2156–2161. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=126323.

[21] Scott L Hopper. *Central Pattern Generator*. 2000. URL: http://crab-lab.zool.ohiou.edu/hooper/cpg.pdf.

[22] Auke Jan Ijspeert. “Central pattern generators for locomotion control in animals and robots: a review”. In: *Neural Networks* 21.4 (2008), pp. 642–653. URL: http://www.sciencedirect.com/science/article/pii/S0893608008000804.

[23] EM Scharf and NJ Mandic. “The application of a fuzzy controller to the control of a multi-degree-of-freedom robot arm”. In: *Industrial applications of fuzzy control* (1985), pp. 41–62.

[24] Guang-Chyan Hwang and Shih-Chang Lin. “A stability approach to fuzzy control design for nonlinear systems”. In: *Fuzzy sets and Systems* 48.3 (1992), pp. 279–287.

[25] Musa Mailah and Numi Izzah Abdul Rahim. "Intelligent active force control of a robot arm using fuzzy logic". In: *TENCON 2000. Proceedings*. Vol. 2. IEEE. 2000, pp. 291–296. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=888750.

[26] Brian Schmalz. *UBW32 (32 bit PIC32 based USB Bit Whacker)*. Mar. 2013. URL: http://www.schmalzhaus.com/UBW32/.

[27] *PICkit 3 Programmer/Debugger User's Guide*. Microchip. 2008. URL: https://www.sparkfun.com/datasheets/Programmers/PICkit_3_User_Guide_51795A.pdf.

[28] *PICkit3 Product Information*. URL: http://www.modtronix.com/product_info.php?products_id=407.

[29] Krodal. *MPU-6050 Accelerometer + Gyro*. July 2013. URL: http://playground.arduino.cc/Main/MPU-6050.

[30] Jun Hee Lee. *General Specifications of HSR-5498SG Digital Robot Servo*. Hitec. 2006. URL: http://www.robotshop.com/media/files/pdf/hitec-hsr-5498sg-digital-servo-specsheet.pdf.

[31] *Lynxmotion SSC-32 Servo Controller*. Robotshop. URL: http://www.robotshop.com/en/lynxmotion-ssc-32-servo-controller.html.

[32] *VLT100 Series: 100W single and quad output AC/DC Power Supplies*. EOS. URL: http://www.eospower.com/All%20Pdf/39-DE60-43931-002A2_VLT100-datasheet.pdf.

[33] *EOS VLT100-4002*. URL: http://www.shopeio.com/inventory/details.asp?id=1711&x=EOS%20VLT100-4002.

[34] *YIHUA 1502DD LED dc power supply*. Yihua. URL: http://yihua-gz.com/Products_detail.asp?id=247&sortid=264.

[35] *Power Supply,YIHUA 1502DD+*. URL: http://gzyihuatools.en.ec21.com/Power_Supply_YIHUA_1502DD--4211637_4216400.html.

[36] Fred R Sias Jr and Yuan F Zheng. "How many degrees-of-freedom does a biped need?" In: *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*. IEEE. 1990, pp. 297–302. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=262401.

[37] Catherine Fiscella. "The Hip Joint". In: *IDEA Fitness Journal* (Mar. 2005). URL: http://www.ideafit.com/fitness-library/hip-joint-anatomy.

[38] Robot Dude. *ssc-32 and max msp*. Lynxmotion Tech Support. Nov. 2010. URL: http://www.lynxmotion.net/viewtopic.php?f=2&t=6867.

[39] R Williamson and BJ Andrews. "Detecting absolute human knee angle and angular velocity using accelerometers and rate gyroscopes". In: *Medical and Biological Engineering and Computing* 39.3 (2001), pp. 294–302.

[40] *8-bit Atmel Microcontroller with 4/8/16K Bytes In-System Programmable Flash*. Atmel. URL: http://www.atmel.com/images/doc2545.pdf.

[41] *High Current DC Power Supplies*. B&K Precision Corp. 2013. URL: http://www.bkprecision.com/downloads/datasheets/179x_datasheet.pdf.

[42] *VRLA Rechargeable Battery BP13-6 S (BP13-6FR S)*. BB Battery Co., Ltd. Feb. 2011. URL: http://www.bb-battery.com/productpages/BP/BP13-6S.pdf.