Abstract
The research and development of robotics has been improving in recent years to the point of being modeled after humans, allowing them to perform tasks as well or better than humans. The purpose of this project is to create the fourth iteration in a series of humanoid robots. This robot will be able walk, turn, avoid obstacles, stay balanced, and pick itself up if it falls. The robot will be untethered, voice-activated, and completely autonomous. The robot is controlled by a Roboard microcontroller, which controls the motorized servos as well as voice recognition, wireless connectivity, and IMU control. An electrical board is used for powering the motorized servos, as well as the microcontroller, which in turn monitors and controls the electronic device components.

Nomenclature
- CAD – Computer-Aided Design
- DOF – Degrees of Freedom
- IMU - Inertial Measurement Unit
- PWM – Pulse Width Modulation
- OS – Operating System
- PCI – Peripheral Component Interconnect
- SSH – Secure Shell
- I2C – Inter-Integrated Circuit
- COM – Component Object Model
- IR - Infrared

Introduction
The ability to create a humanoid robot has been a long-desired goal in the field of robotics. A robot designed after a human would ideally be capable of carrying out complex tasks that current machines are unable to do. A humanoid design is difficult to implement, as there are issues of maintaining balance and posture, as well as delivering enough torque to power the legs while the robot is standing. This TigerBot iteration attempts to advance the solution for these problems and create a robot with the main goal of automated walking.
This project is the fourth iteration in a series of humanoid robots, stemming from a third iteration (P13201), a second iteration (P12202), and the original project (P12201). P12201 served as a starting point, resulting in a humanoid design but being unable to walk. P12202 used a new humanoid design that was lighter, but ultimately the amount of force on the lower half proved to be too much for the servo motors used. P13201 used servo motors with a higher torque, but the weight of the design prevented the robot from being able to walk on its own. This design uses techniques from the previous iterations by creating a mechanical framework that is sufficiently light yet free of flexing and joint wobble, as well as using high torque servo motors in the lower half where necessary, while using servo motors with a lesser torque in the upper half. The result is a humanoid robot that is optimized for size and weight, which should help in the process of the robot walking on its own.

**Design process**

![Mechanical CAD](image1.png) ![Joint Bearing Design](image2.png)

**Figure 1 – Mechanical CAD**  
**Figure 2 – Joint Bearing Design**

**Mechanical Design:**

The mechanical portion of the robot called for creating a humanoid design, as seen in Figure 1, and optimally allowed for 23 DOF. To create the frame, 1/8” aluminum was used for the majority of the robot. This metal made up the joint brackets and as well parts of the torso and the feet. To connect each part of the robot, hollow aluminum rods were used, which helped to reduce the amount of weight in the overall design. This single rod limb design also lends to a more modular design as limb lengths can be easily changed by installing different length rods at each location.

To design the shoulder, elbow, and knee joints, the servo was mounted inside a bracket assembly, as seen in Figure 2. All Y and Z loading in the joint goes through the two radial bearings on the sides of the bracket, while the servo provides only the rotational torque for the two limbs relative to one another. To ensure that no part of this rotational assembly takes any non-rotational load, the mounting pins of the servo gear to Limb B should not be constrained in X or Y, but constrained only in the rotational direction (about X). If it is found that a joint has an axial load at any point in its motion (due to robot posture for certain maneuvers), there is flexibility to add a thrust washer or bearing to each side of this joint between the two limbs, taking up the axial load (X direction in this sketch). In the actual implementation during the build, thrust washers were installed at these locations for every joint. Also, if there is any issue with looseness in this joint in the axial direction, X, there is the ability to add a fastener through the left side bearing, constraining Limb A and B relative to one another in X. This was found to be unnecessary during the build as the bearing press fits were enough to constrain the joints in X.
To connect the upper half to the lower half of the robot, it was necessary to design a pelvis that could connect the two parts, as seen in Figure 3. All Y loading from the Torso to Pelvis, and Pelvis to each hip joint, goes through thrust bearings, while the servos provides only the rotational torque for the two hip joints relative to the pelvis. To ensure that no part of this rotational assembly takes any non-rotational load, the mounting pins of the servo gears should not be completely constrained, as explained in Figure 2. In order to establish the final two rotational degrees of freedom required in the hip, a housing box will be used. Each box will contain the two servos, as well as complete bearing setups as previously explained, and will take the entire load between the intermediate bracket and the limb below through these two sets of bearings. This housing box setup can also be used at other joints which require two degrees of freedom in a tight package, such as the shoulder and ankle. The housing box itself can be seen in Figure 4, and was used for the hip and ankle design.

**Shell Design:**

The robot is covered with an exterior shell, which helps to hide some of the internal components and make the robot more aesthetically appealing. The shell is made out of a thin metal, which is shaped to resemble Iron Man. The design for this shell was created by using Pepakura, which takes 3D data and allows it to be crafted into an actual model. This model was then placed onto the metal, which could then be cut out as desired. The individual pieces of metal are connected to each other with rivets, which hold the metals in place. The shell encompasses most of the mechanical and electrical portions of the robot, and can be easily removed to allow access to the inner portions. Along with allowing for ease of access, the shell design also does not interfere with servo movement, allowing the robot to keep the same movements.
Software Design:

The software flow for the robot followed a pattern that would keep it moving autonomously as well as protect the internal components, as seen in Figure 7. Once the robot was powered on and set to stand, the first step would be to determine if it has fallen over. If so, all movement would be stopped to prevent any potential damage to the servos. The robot could then proceed to pick itself up. The next step would be to look for an object, and stop movement if one was detected. If no object was detected, the software would check for a new command, and carry it out. Otherwise, the robot would just maintain its current action. If no such action was taking place, the robot would simply maintain its balance, going through the main loop once again.

Electrical Design:

The electrical design accounted for each internal component being powered in two different manners, as seen in Figure 8. The power board was powered by two batteries, and could then be used to power the servos and the logic components separately with switches. This separation allowed the robot to be safely powered on without activating any movement, and also allowed for the servos to be safely shut off without turning off the rest of the robot. The servo power would simply activate the servos in the robot, allowing them to move as desired. The logic power would power the current sensing board, the Arduino, the IMU, the Roboard controller and its separate components. The Robo board controller would supply enough power to the IR sensors, voice recognition, wireless board, and camera, allowing them to operate normally. The Robo board controller would also be used to connect to the power board through an I2C master, which could then be used to control separate I2C devices such as the IMU and current sensing from the Arduino.
Microcontroller:

The microcontroller selected for this design was the Roboard RB-100, as seen in Figure 9. This controller was selected primarily for having 24 PWM pins, which allowed for independent control of all 21 servos in the robot. The Roboard controller is also capable of running its own OS off of a microSD card, which was used to run a custom version of Lubuntu [1], which is a light version of Linux. This allowed for the main portion of programming to be done directly on the controller itself, allowing for the robot to be run autonomously. Along with servo control, the Roboard controller can also run a number of peripherals, including wireless connectivity, an IMU, and voice recognition.

Servos:

The servos used for this design vary depending on its location in the robot. For the upper half of the body, Roboard RS-1270 servos were used, which were the same type of servos used in P12202. These were found to generate enough torque to power the upper limbs, as well as the pelvis area. For the lower limbs, however, more torque was required to move the servos to the desired positions, given that the humanoid design of the robot put most of the weight on the lower half of the body. Therefore, XQ-S5650D servos were used for the hip, knee, and
ankle portions of the robot. These servos produced a higher amount of torque than the Roboard servos, which in turn generated enough force to move the legs as desired. Finally, two Hitec HS-422 servos were used for the head, which allowed for turning the head up, down, left, and right. These servos were chosen for their small size and cheap price, as a lower quality servo was all that was necessary for moving the head.

**IMU:**

The IMU for this project contains a gyroscope and accelerometer, which help determine whether the robot is standing or has fallen. This board is placed in the middle of the robot, and is then connected to the Roboard controller through an I2C connection. When the IMU is running, the robot will be able to tell how its current state of balance, so that if the robot has fallen or is about to fall, the Roboard controller will know to power off its servos and go into a protective position.

**Wireless Connectivity:**

The Roboard contains a PCI slot which allows for certain components to interact with the Roboard. For this project, a custom wireless VNT6655 card was connected to the Roboard. The custom OS installed on the Roboard would recognize the wireless card’s hardware, and the Roboard was configured to connect to RIT’s wireless network. Once the card connects to the network, a remote computer can connect to the wireless card by using SSH, which will then allow the Roboard to be controlled directly from a remote location.

**Power Board:**

To power the robot, a custom power board was designed to handle controlled power to the servos, microcontrollers, and other peripherals. The power board is powered through two rechargeable Tenergy 7.2V 3800mAh batteries, allowing for the robot to be untethered. The board has two switches attached, allowing the user to enable or disable power to the logic (microcontroller) portion of the robot, as well as to the servos on the robot. If a battery is found to be low, a low-battery light will turn on to let the user know.

**Current Sensing:**

If one or more servos is using too much power, they are prone to breaking or moving the robot into an unsafe position. If the current for each servo is known, then appropriate actions can be taken to adjust the servos if necessary. To determine the amount of current being drawn by each servo, a current sensing board was designed to connect to each servo and determine how much current each one was using. To handle this information, the current sensing board was connected to an Arduino Mega 2560 microcontroller to process this information. The Arduino was used as it could handle up to 16 analog inputs, while the Roboard controller could not handle as many analog inputs. The Arduino was then connected to the Roboard, allowing appropriate servo actions to be taken.

**Inverse Kinematics:**

To make the robot walk, inverse kinematics were used in the microcontroller’s software to calculate the angles of the robot’s lower limbs. A series of equations were generated in Matlab, which produced a lookup table that is used by the microcontroller. The current angles of the lower limbs are inputted to these equations, which would determine what position the limbs should go to. Once these exact positions were determined, the microcontroller could move the servos to these positions, allowing for the robot to complete a walking motion while still being able to balance itself.

**Voice Recognition:**

One requirement for this project is to allow for voice recognition, so an EasyVR board was used to take a voice input from a user and process it. This board was connected to the Roboard controller through a COM interface, which allowed it to communicate what sounds it had processed. The EasyVR board has a microphone which can pick up a user’s voice, and comes previously programmed with a set of commands that allow for recognition from any person. Once a recognized command is received from the user, the EasyVR board sends this information to the Roboard controller, and the controller determines what command to carry out based on what was received.

**IR Sensors:**

To allow for object detection, a group of IR sensors was placed on the robot at various points. Each sensor determines how far away an object is by sending its information to the microcontroller. If an object is found to come within a certain range of any of the sensors, the microcontroller will stop movement in the servos, effectively stopping the robot so that it will successfully avoid hitting any obstacles.
Results and discussion

A majority of the objectives were completed by the end of the project. The Roboard CPU was able to communicate wirelessly with the RIT network, and was able to send pictures from its webcam wirelessly as well. The robot was able to receive and respond to certain voice commands, moving to pre-programmed positions. The robot was able to balance itself in a stationary standing position while powered on, able to resist falling by a small amount. The robot could move autonomously while still maintaining balance, and was able to recognize any obstacles and be able to stop movement. The overall mechanical structure was strong as designed, and still achieved a low weight of 18.5 pounds, which was about 5 pounds lighter than the previous TigerBot iteration. A percentage of weights for each component of the robot can be seen in Figure 10.

Due to time constraints, it was not feasible to complete every requirement. While the robot was able to stand and balance by the end of the project, there was not enough time to implement and test a proper walking algorithm. Given that the robot was able to stand and balance, along with the implementations of inverse kinematics and current sensing, the robot should be mechanically able to walk. If given more time to implement this feature, it could definitely be functional for this project iteration. Along with the walking requirement, the robot was also not programmed to pick itself up from a fall. The main focus of the project was on having the robot balance itself while standing and walking, so this objective was given a lower priority. The robot was simulated to pick itself up from a fall, so it is possible that the actual robot would be able to do this if given enough time. A final problem of the time constraint was the fact that testing of the robot standing and walking could not be done until the entire robot was assembled, which gave less time than desired for the software portion of the project.

<table>
<thead>
<tr>
<th>Number of Components</th>
<th>Actual Weight (g)</th>
<th>Actual Total Weight (g)</th>
<th>% breakdown of weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servos (Hi-Tec)</td>
<td>2</td>
<td>45.36</td>
<td>90.72</td>
</tr>
<tr>
<td>Servos (Roboard)</td>
<td>11</td>
<td>70</td>
<td>770</td>
</tr>
<tr>
<td>Servos (XQ)</td>
<td>10</td>
<td>177</td>
<td>1770</td>
</tr>
<tr>
<td>Structure Materials</td>
<td>multiple</td>
<td>4240.24</td>
<td>4240.24</td>
</tr>
<tr>
<td>Batteries</td>
<td>2</td>
<td>382.72</td>
<td>765.44</td>
</tr>
<tr>
<td>Electronics Components</td>
<td>multiple</td>
<td>755.06</td>
<td>755.06</td>
</tr>
<tr>
<td><strong>Total Weight</strong></td>
<td></td>
<td><strong>8391.46g = 18.5lbs</strong></td>
<td></td>
</tr>
</tbody>
</table>

Conclusions and recommendations

A large majority of the objectives were completed at the end of the project, and this iteration of TigerBot is seen as the most successful of all iterations. The size of the team seemed just about right, with more than one type of engineer in each group, except for ISE. As stated previously, it was difficult to divide the work to the software portion of the project until the end when the actual robot was assembled. It would be possible to work around this by working with a previous iteration of TigerBot instead of starting from scratch, or by extending the amount of time allotted to the project.

A few recommendations could be made for further work on this project. First, a budget higher than $2500 would be necessary. This value could be nearly met simply by purchasing the servo motors, which make up a large portion of the project and are the most vital. Second, a new form of voice recognition should be considered, as the module used for this project did not work well from a large distance, and could likely be better implemented in a phone or tablet application. When working to program the robot, the batteries powering it will require time to charge, so work should be spread out to accommodate for this.

From a mechanical standpoint, future teams would be advised to avoid designs that require bending brackets. The bending press equipment at RIT has inaccuracies that led to a large amount of build corrections requiring welding and a couple weeks of rework. The pinning of the servo output shafts was also the largest area of concern with the mechanical design. As RIT does not have the capability to machine internal splines, the servo horn mounting is an area that warrants a great deal of focus during the design phase. It is also advisable for the mechanical team to meet with the machine shop lab technicians during MSDI to review design and fabrication ideas to see what can and cannot be done.
Acknowledgments

Our team would like to thank our sponsor, Dr. Ferat Sahin, our guide, Prof. George Slack, and Dr. Becker-Gomez for their help and guidance throughout the project. We would also like to thank Roboard for their contributions. Also, a special thanks to the machine shop technicians, Rob Kraynik and Jan Maneti, for their help and guidance throughout the mechanical build. Finally, we would like to thank the members of the third P13201 TigerBot team for their direct assistance with this project.