Michael Yau Chan, Daniel Jarc, Jacob Powers
CS467-400 Capstone Project
January 20, 2019

Team Fang Project Plan

Introduction

Our team will be building and programming a 3D printed, quadruped robot using C++ and an Arduino microcontroller. The quadruped will have 3 degrees of freedom on each leg and be capable of walking, body rotation, tilt and translation. The robot will interface with custom control software connected via Bluetooth.

Team Members

Michael/Yau Chan - chany@oregonstate.edu
Daniel Jarc - jarcd@oregonstate.edu
Jacob Powers - powersj2@oregonstate.edu

User Perspective

As a user of the hardware and software that make up the robot I would expect that the robot would operate in an manner persistent with its design. The robot requires the harmonious operation of many different systems: hardware, movement and controls. The movement software will be the centerpiece of the robot and will tie all various hardware components into a single functioning machine. The inner workings of the code and relationship of parts is hidden from the user. The user will be presented with basic controls as outlined in the control software to operate the movements of the robot.

Client

Our client is Ryan Gambord and/or the TA who will be grading this project. In addition, this extends to anyone interested in robotics. The design of our projects is simple yet extensible enough that it can be built and improved upon by anyone. The parts are all cheap and readily available

Structure

The project can be broken down into several primary sections - movement system, control system and frame design/hardware. The movement systems will consist of code 3rd party libraries as well as our own libraries.

Movement System

The movement for the quadruped will be written using the Arduino IDE using C++ and the open source Arduino libraries. The body and leg movement algorithms will need to utilize inverse kinematics. Initially, the following capabilities will be developed:

- [ ] Tilt body, stationary
- [ ] Rotate body, stationary
- [ ] Translate body horizontally, stationary
- [ ] Translate body vertically, stationary
- [ ] Translate/walk in desired direction

- ❏ Rotate while walking

If these primary movement mode goals are completed, additional movement gaits and functionality will be added, possibility including additional walking gaits, scripted sequences, simple obstacle detection. The Adafruit servo library will be used to control leg servos.

The movement code will be encapsulated within a quadruped class:

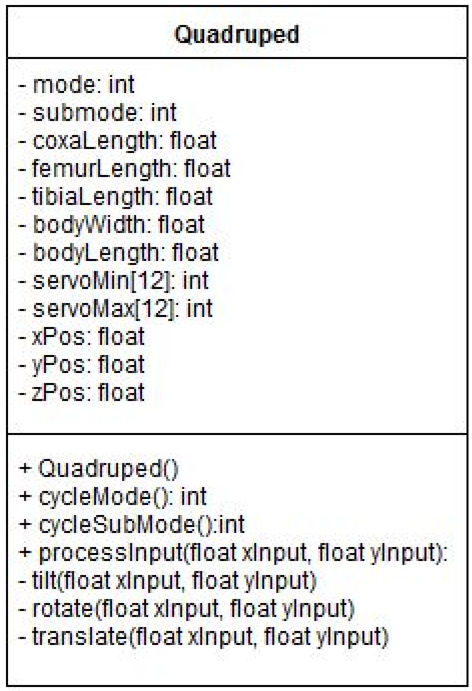


Figure 1: Quadruped Class

The Quadruped constructor will take arguments pertaining to the frame dimensions and servo calibration settings. The user will interface with the quadruped by setting a mode (default: stationary) via cycleMode(), submode (default: tilt) via cycleSubMode() and process directional input from the virtual joystick via processInput(). The processInput() function will move the quadruped depending on the currently selected mode and submode.

Control Software

The quadruped user interface will be designed using the Qt 5 framework and C++. The software will consist of a single screen interface with the following functionality:

- ❏ Mode Select - Cycles between walk and stationary modes
- ❏ Submode Select - Cycles selected mode submodes
- ❏ Connect - Connects to quadruped via bluetooth
- ❏ Status - Indicates connection status
- ❏ Directional pad/joystick interface – Controls quadruped according to mode and submode selections

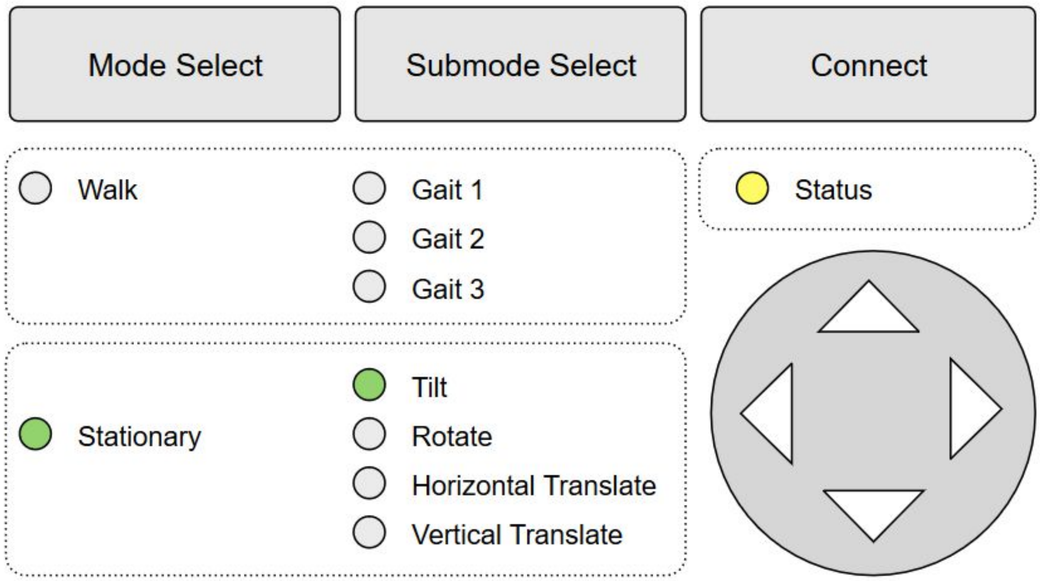


Figure 2: UI Mockup/Layout

Applicable [Qt modules]include:

- QtCore - Provides core non-GUI functionality
- QtGui - Basic enablers for graphical applications
- QtWidgets - Extends Qt GUI with C++ widget functionality
- QtBluetooth - Enables basic Bluetooth operations like scanning for devices and connecting them

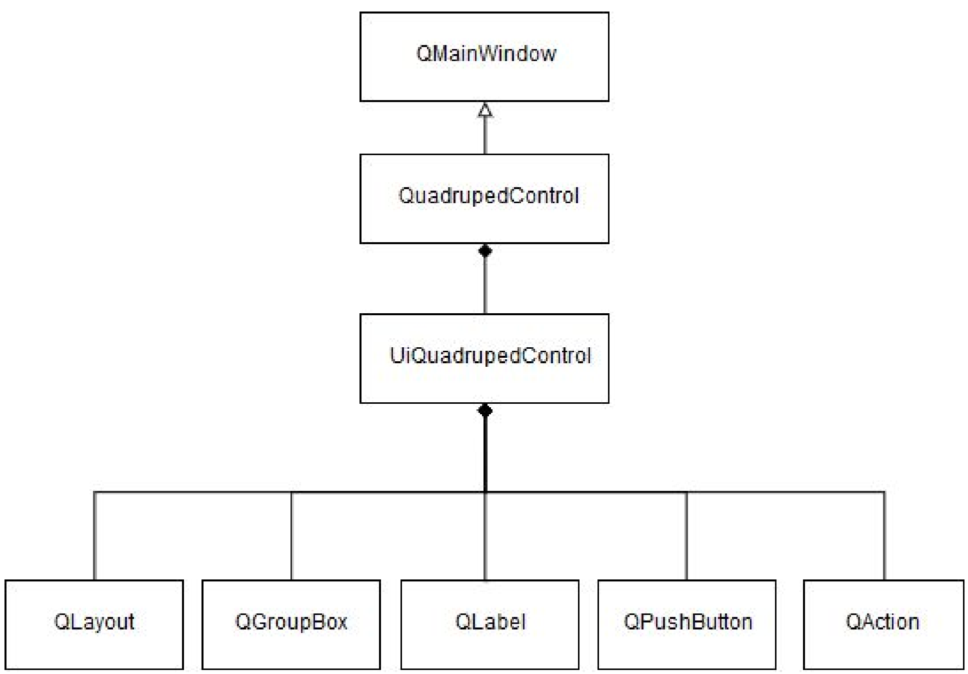


Figure 3: App class diagram

The *QuadrupedControl* class will contain all the logic/functionality of the app while the *UiQuadrupedControl* class will contain all the visual components. The UI will be connected to the logic via slots and actions.

Arduino Bluetooth Hardware Integration

The Arduino [SoftwareSerial](#) library and a HC-06 bluetooth module will be used onboard the Arduino. The bluetooth connection process is illustrated below.

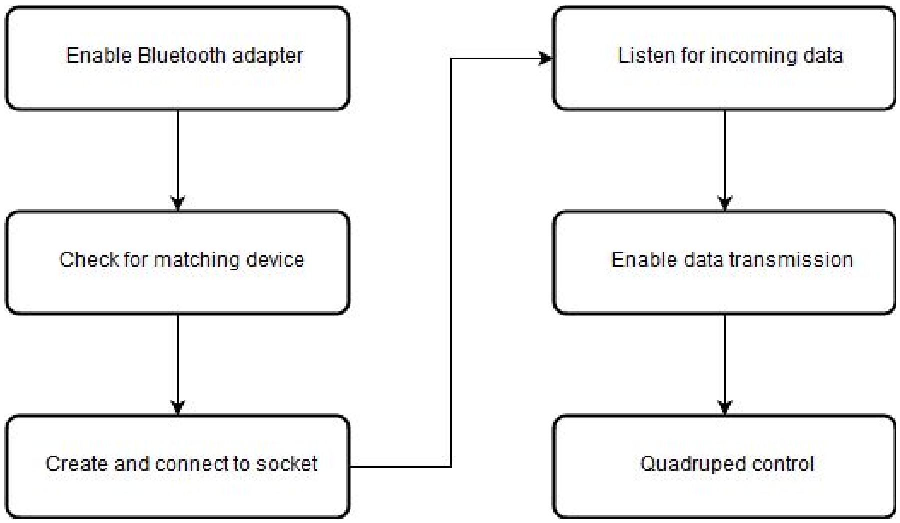


Figure 3: Connection Process

Frame Design

A preliminary frame design was created using Autodesk Inventor and a Monoprice Mini Delta 3D printer. Below is a screenshot from Inventor displaying the frame components. This frame is a work in progress. Modifications and refinements to this design will occur throughout the development.

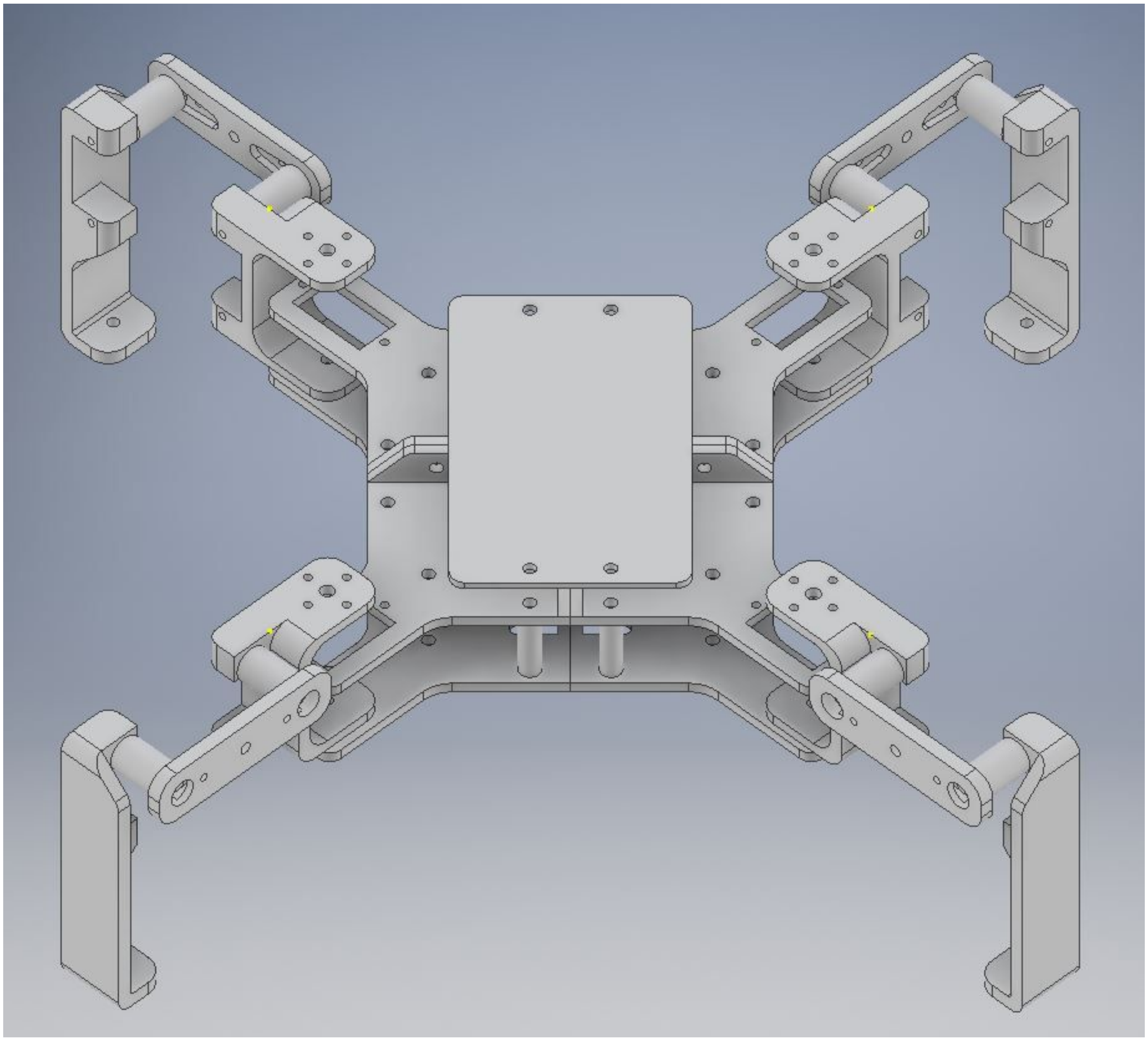


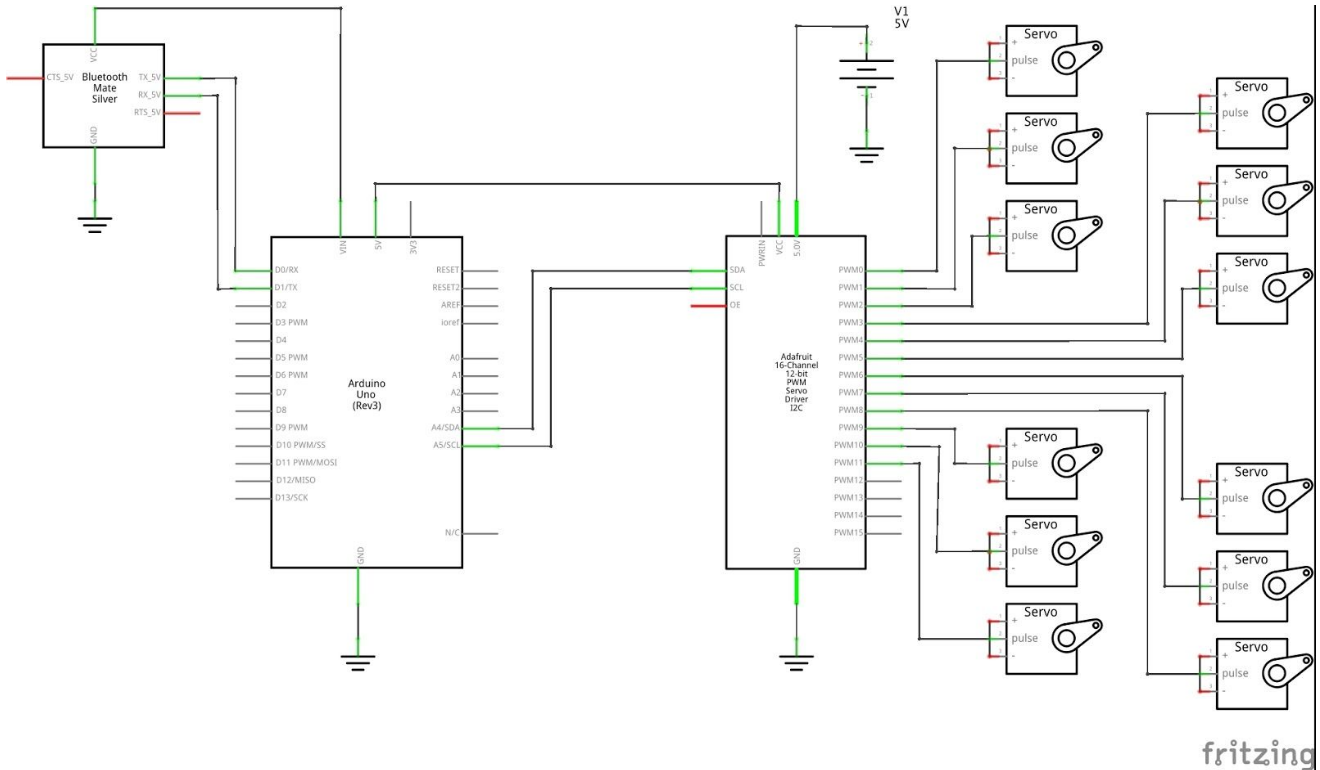
Figure 4: Prototype Frame Design

Build Components

1x Arduino Uno microcontroller
1x Adafruit servo shield
1x 2S 1000 mAh Lipo or 4X AA Battery (servo power)
1X 9V battery and barrel plug adapter
1x 3A 5V UBEC
12x SG90 servos
1x Bluetooth module
2x Upper/lower body assemblies
4x Shoulder assemblies
4x Femur assemblies
4x Tibia assemblies
4x 2 mm ball bearings

Assorted M2, M3 screws, washers, nuts, nylon spacers

Figure 5: Wiring diagram



Software Libraries

- Arduino IDE
  - Quadruped movement system
- Qt Creator
  - User interface/control software
- Adafruit servo library
  - Servo control
- SoftwareSerial library
  - Bluetooth connectivity
- Qt Widgets library
  - UI elements/controls
- BasicLinearAlgebra Library
  - Arduino math library

Additional open source libraries may be used depending on development needs. The primary language used will be C++ for both the robot code and UI code.

Plan

Below is a high-level plan of the tasks required to complete the project and our target completion dates. Our group will utilize Slack for messaging and Google Hangouts for weekly meetings (Sundays).

Task	*Target Completion*
Servo alignment	Jan 11
Final frame assembly	Jan 23
Stationary - tilt body	Feb 1
Stationary - rotate body	Feb 1
Stationary - translate body horizontally	Feb 1
Stationary - translate body vertically	Feb 1
Move - translate/rotate	Feb 8
Control software/UI	Feb 15
Bluetooth module integration with UI	Feb 22
Testing, debugging, improvements	Mar 17

Class Assignments

Below is a summary of the required class assignments and due dates.

Assignment	*Due Date*	*Assignee*
Week 4 Progress Report	Feb 1	Individual
Week 5 Progress Report	Feb 8	Individual
Mid-Point Project Check	Feb 15	Jake
Week 7 Progress Report	Feb 22	Individual
Week 8 Progress Report	Mar 1	Individual
Week 9 Progress Report	Mar 8	Individual
Create Poster	Mar 17	Dan Jarc
Create Final Report	Mar 17	Michael Chan
Demonstrate Project	Mar 17	Everyone

Individual Task Assignments

Below is a detailed plan of individual task assignments, organized by week.

Team Member: Michael/Yau Chan

Week	Tasks	Time
3	Project plan and tools ramp-up	10
4	Create preliminary UI with Qt	10
5	Create preliminary control software to interface with Qt	10
6	Mid-Point Project Write-up	10
7	Create preliminary bluetooth control to interface with Qt	10+
8	Combine control software and Qt UI, look into additional features if time permits	20+
9	Testing and debug	20+
10	Poster and final report	10+

Team Member: Daniel Jarc

Week	Tasks	Time
3	Frame prototyping, project plan, research	10
4	Final frame assembly, single leg IK, research	15
5	Stationary modes (tilt/rotate/translate)	15
6	Basic movement, motion control	15
7	Additional gaits, modifications, motion control	15
8	Control software/bluetooth integration	15
9	Testing, debugging	10
10	Poster, final report, demonstration	20

Team Member: Jacob Powers

Week	Tasks	Time
3	Research robotic kinematics, Initial bluetooth, Servo calibration	15
4	Frame assembly, single leg IK, Research, Test development	15
5	Stationary modesm, Test	15
6	Basic movement, Test, Midpoint writeup	15
7	Additional gaits, Test	15
8	Control software/bluetooth integration, Test	10
9	Testing, debugging	20
10	Poster, final report, demonstration	

Conclusion

This project will allow team members to explore embedded systems, inverse kinematics, bluetooth connectivity and user interface design. In addition, this project allows us to use the software methodologies and programming techniques that we have been developing over the course of the OSU CS post-baccalaureate program into a concrete application that can be tested and used by others.