EECS 581 Group 2 Final Project Design

Team Name: STV Robotics
Team Number: Team 2
Team Members and email addresses:

1. Sri Gayatri Sundara Padmanabhan, sri@ku.edu
2. Paul McElroy pcm@ku.edu
3. John McCain johnm.freestate@gmail.com
4. Luke Dercher luke.dercher@gmail.com

Team Meeting time: Monday 1:45 pm
Lab Meeting time: Wednesday 2:00 pm
Contact:

- Paul McElroy: phone # (785) 215-1633
- Luke Dercher: phone # (228) 313-5570
- John McCain: phone # (785) 218-5993
- Sri Gayatri: phone # (408) 963-7928

Project Sponsor (if any): n/a

<u>Project Description (150-250 words)</u>

- Why is the project being undertaken? Describe an opportunity or problem that the project is to address.

To enable people with basic programming skills to build and configure a telerobotics platform utilizing a web-based interface with the ability to connect and control. Applications of our platform include security cameras, medical robotics, industrial robotics, telepresence, and more. Our platform will allow robotics hobbyists to take their projects to the next level, businesses with limited budgets to build custom telerobotics solutions at a lower cost, loved ones to interact over long distances, and allow employees to work remotely easier.

- What will be the end result of the project?

The end product of our project will be a telerobotics platform with minimal setup that anyone can use. Clients can connect from a remote device such as a laptop or mobile device to a central server through a web application. The server will then connect clients with the available registered robotic agents. Depending on the agents available and their capabilities, different configurations will be available for the users to select which will define the user interface for the robotic agents. Users will be able to configure basic UI for any connected robot that implements our system. We intend to complete the project with an example robot & configurations.

<u>The end result of this project</u>

The end product will consist of a web based platform that exposes a robot facing API for communication and control, a client facing REST api for configuration and control, and a SPA style web application that uses the client facing REST API and allows for user control of the whole platform.

Project Milestones

First Semester Goals

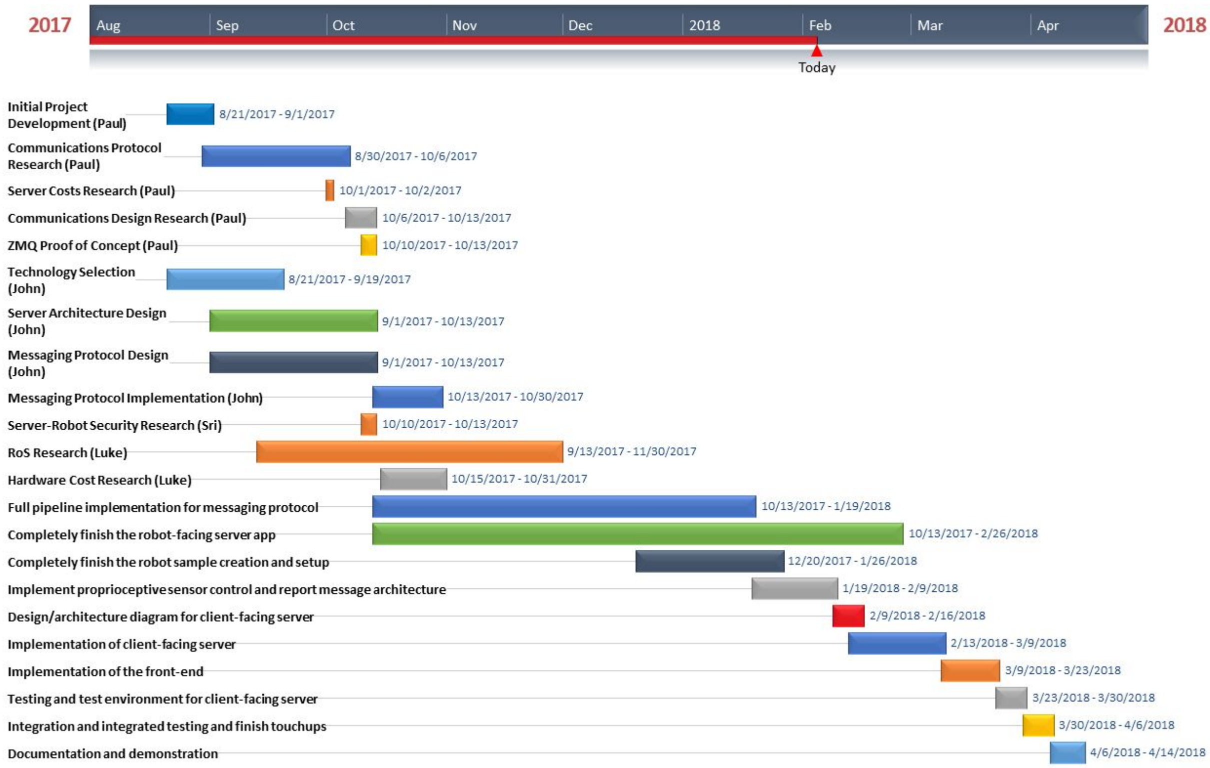
- Design and initial documentation of robot-server messaging protocol (Oct 23rd)
- Proof-of-concept implementation of robot-server messaging protocol (Nov 6th)
- Design and creating of the sample robot (Dec 27th)

Second Semester Goals

- Design and proof-of-concept implementation of core server-side services (Feb 28th)
- Proof-of-concept level demonstration of core platform functionality from client REST API to robot communication (March 15th)
- Full implementation and documentation of client side REST API (Mar 26th)
- Front end control application implementation (March 30th)
- Demonstration of robot-user connection from front end application to robot communication (April 9th)
- Final documentation and release-ready product (April 14th)

Note: documentation and test coverage will progress alongside development

Gantt Chart



Project Budget

- Hardware, software, and/or computing resources
  - Lynx motion johnny 5 robotics kit: $879.99 http://www.robotshop.com/en/lynxmotion-j5c-kt-johnny-5-kit.html
  - Pan tilt kit for robot head: $29.93 http://www.robotshop.com/en/lynxmotion-pan-and-tilt-kit-aluminium2.html
  - Robot shocks: $15.99 http://www.robotshop.com/en/actobotics-aluminum-robot-shocks-pair.html
  - Blackbird 1 3D FPV Camera: $89.00 http://www.robotshop.com/en/blackbird-1-3d-fpv-camera.html
  - 6V battery: $26.95 http://www.robotshop.com/en/6v-2800mah-nimh-battery.html
  - 12V Battery Pack: 48.99 https://www.robotshop.com/en/120v-2800mah-rechargeable-nimh-battery-pack.html
  - Raspberry Pi 3 Ultimate Kit: $102.67 https://www.robotshop.com/en/raspberry-pi-3-ultimate-kit.html

- Generic M2.5 Nylon Hex M/F Spacer/Screw/Nut Assorted Kit, for Raspberry: $11.99 https://www.amazon.com/Generic/Spacer/Assorted/Raspberry/Pi/Standoff/dp/B014J1ZLD6
- Monoprice Feet USB 2.0 A Male to A Male 28/24AWG Cable ;Gold: $4.25 https://www.amazon.com/Monoprice-1-5-Feet-24AWG-Plated-105441/dp/B009GUXG92/ref=sr_1_15?s=wireless&ie=UTF8&qid=1508887105&sr=1-15&keywords=usb+cable
- 12V NiMH / NiCd Smart Charger: $21.95 http://www.lynxmotion.com/p-602-6-12v-nimh-nicd-smart-charger.aspx
- ZILU Smart Power Basic 4400mAh Portable Charger External Battery Pack Backup: $14.90 https://www.amazon.com/gp/product/B00MWV1TJ6/ref=as_at?creativeASIN=B00MWV1TJ6&linkCode=w50&tag=mak041-20&imprToken=XkZOfOfTNscHdyVfluncMA&slotNum=1
- Wiring Harness; Battery Connector: $4.95 http://www.lynxmotion.com/p-497-wiring-harness-battery-connector.aspx
- Standard Servo: $16.29 http://www.lynxmotion.com/p-719-hs-485hb-83-oz-in-standard-servo.aspx
- ZIYUN,Sabertooth Dual 5A DC Motor Driver: $136.31 https://www.amazon.com/ZIYUN-Sabertooth-currents-achievable-differential/dp/B076Q58KGL/ref=sr_1_14?ie=UTF8&qid=1517435916&sr=8-14&keywords=sabertooth+motor+driver
- Server Hosting by KUIT: Free

- **Estimated cost**
  - Total robot cost: $1361.27 + tax

- **When they will be required?** Already received all parts.

Work Plan

- Luke Dercher (SCRUM master) will be taking over the creation of the robot software layer in the CoE team's absence. A robot kit with minimal hardware assembly will be used instead of the one intended to be provided by the ex CoE team.
- Sri will work on building the server backend and front-end, security issues, and will pitch in and help other teammates with other aspects of the project.
- Paul will be working on the design and implementation of Video services, frontend and backend implementation and will be assisting Luke on implementing his work on the robot software layer.
- John will work on system architecture, back end development, and design and development of the front end application

Github link
We are storing our code in multiple repositories within a single Github organization. All repositories will be publicly viewable and open source. We are using Trello and Slack for planning and communication. Here is the link to our GitHub page.
https://github.com/KUSeniorDesignWebRobot

Final Project Design

The platform will exist in three primary domains: a robot layer built on top of Debian, a server layer, and a front-end web application.

The server layer will handle the transmission of messages between the robot and web client, message translation, authentication, data storage, and video processing. It will be based on a service oriented architecture to allow for horizontal scalability and easier testing and development. There will be three publicly exposed services, a web client interface, a robot client interface, and the video processing service (if time applicable). All other services will be accessible through internal DNS to limit potential attack surfaces for security. Internal communication will happen through HTTP, TCP/UDP, and a RabbitMQ message broker. The message queue will handle all messaging which does not require a prompt response, including but not limited to messages between the web client and robot during processing and diagnostic messaging. RabbitMQ is a scalable implementation of AMQP (Advanced Message Queuing Protocol) with clients available for most commonly used languages. Using a pub/sub message queue for internal communication rather than pushing messages with HTTP or TCP allows for tolerance of moments of high load and the potential for automatic scaling during peak usage.

Other technologies we will use on the server include PostgreSQL, Redis, and NodeJS. PostgreSQL is a feature rich relational database. PostgreSQL offers indexed semi-structured data document storage functionality, which we will use in lieu of a dedicated document storage engine like MongoDB for caching, schema-less document storage, and storage of immutable data which lends itself to a key-value retrieval method. All other long term data storage will be stored in a relational structure. We will also use Redis, a primarily in-memory key-data structure store for transient or volatile document storage which requires high speed access.

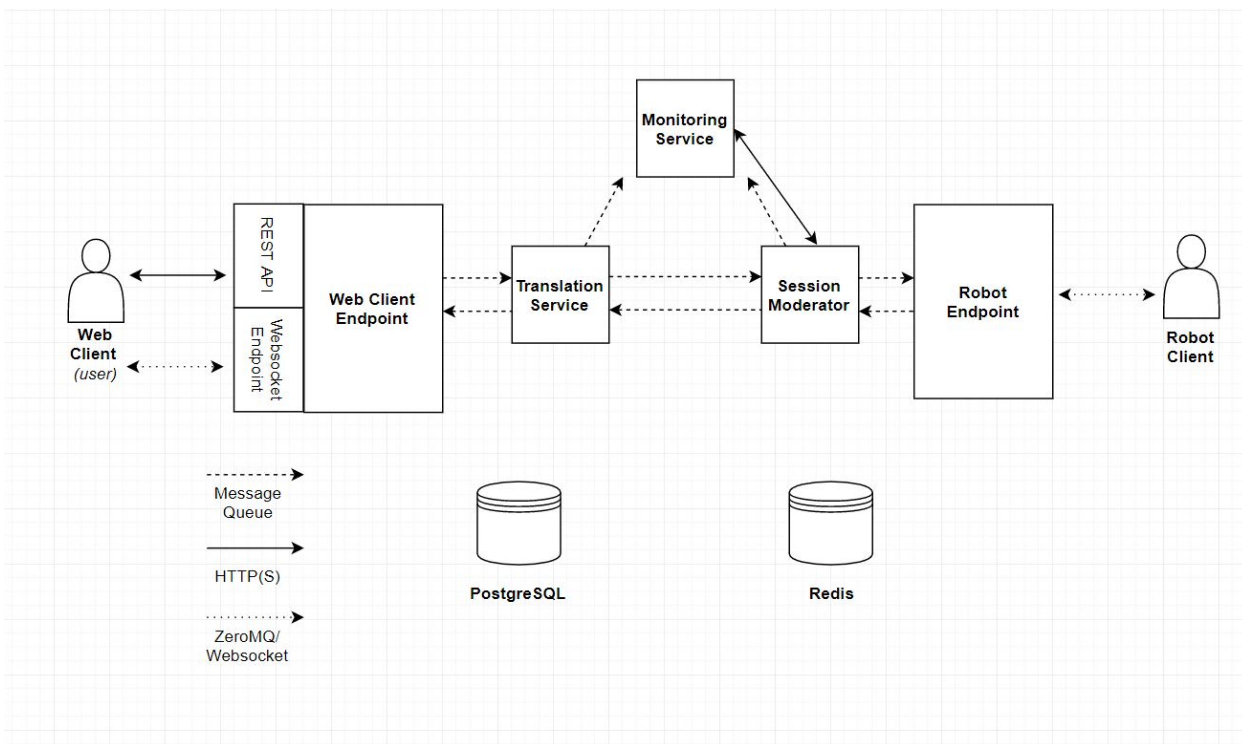
Our services will be written primarily in NodeJS. NodeJS is particularly performant in IO bottlenecked tasks, which we anticipate will be the primary performance restriction. Node is also easy to rapidly prototype with and has impressive speed for an interpreted / just in time compiled language. Its single threaded nature is not limiting due to our plans for a horizontally scalable system, as we will be able to spin up more worker nodes for most services as needed.

We will use Websockets for communication with the Webclient and ZeroMQ for communication with the Robot. Websockets are a secure standard for full duplex communication between a web browser and a server with good levels of modern browser support. ZeroMQ is a brokerless, secure message protocol which works over TCP and UDP.

ZeroMQ can handle large message sizes (including video) at high speed and has clients available for many common languages.

Our service oriented architecture design will likely require additional services that we will not realize the necessity of until later in the development process, but a few of the services we plan on implementing are:

- REST API for web client (includes much of the traditional web app type functionality)
- Translation service (for translating user input into the required format specified by the robot's manifest)
- Monitoring service (tracks latency metrics and session statistics)
- Session moderator service for setting enforced latency and managing the creation and termination of sessions
- Robot facing endpoint
- Web client facing websockets endpoint for message passing



[1]

We originally planned on implementing a live object detection system using the YOLO framework, but we removed this goal due to time constraints.

The video bandwidth expected is around 8 to 15 fps. For higher FPS we can enjoy more control over how the "users" interact and get feedback from the robots. For lower FPS, however, we will have to include limiters in the functionality of the robots from the "user's" perspective in order to maintain a clean control of the program and hardware. We don't yet

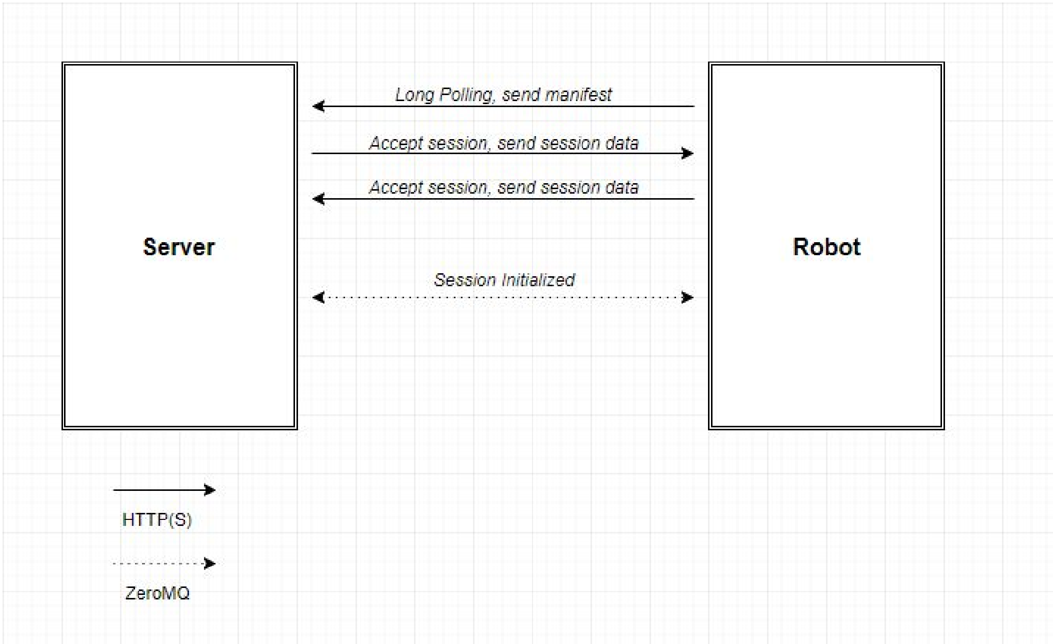
---

[1] *High level software architecture diagram*

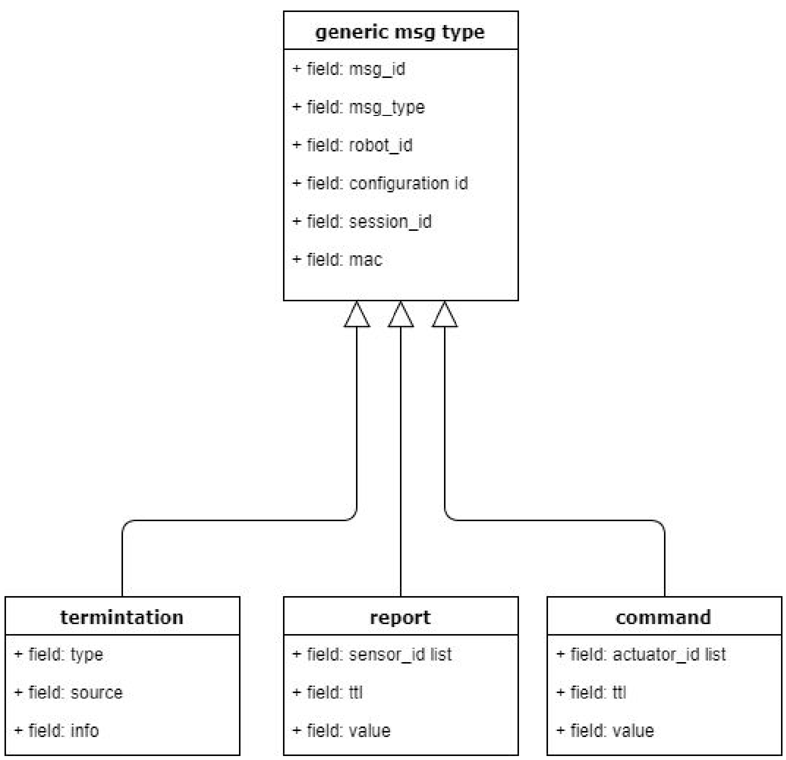
know what kind of latency to expect regarding video and this is a major point of research we will have to undertake to get a workable model for our devices.

We have devised our own message format for communication with the robot. The messages will be passed in JSON. Each message has a timestamp for when it was issued and a variety of metadata, including unique identifiers for all relevant constructs. The primary message types are report and command messages which give sensor information and instruction information respectively. These messages include a list of sensor/actuator unique identifiers, a time-to-live, and a value. Multiple instructions or sensor measurements can be sent in a single message given that they are issued with the same timestamp. Communication will be session based, use public-private key encryption, and based on a request-reply pattern within ZeroMQ for security purposes. Messages to the robot will have an enforced latency in order to keep the latency of commands consistent. We believe that longer but predictable and stable latency is preferable to shorter but inconsistent latency when it comes to robotic control. The timeline of a session is as follows:

- Powered on and web connected robot enters a standby mode, long polling the server to indicate availability. During this phase the robot sends a manifest document to the server. The manifest contains a complete list of all registered actuators and sensors along with descriptions and accepted value ranges.
- The server replies to the robot and asks to initiate a session when an authorized user starts a connection.
- The robot accepts the request to begin a session and sends necessary session information such as public keys necessary for authentication.
- The server attempts to recall a configuration that matches the manifest requirements if one exists. If a matching configuration does not exist, the user is prompted to create one.
- The session is initiated, and communication and control begins.



2



3

[2] *Robot-server session initialization process*
[3] *UML Diagram of the robot message protocol specification*

The front end web client application will allow for configuration and control of our platform. It will be built with AngularJS as a Javascript framework. It will be built in the style of a Single Page Application (SPA), and will interact with the REST API endpoint for all actions except robot message passing. This design keeps with our overall design philosophy of separate coordinating applications/services. We will use various web technologies such as Babel, Passport, Gulp, SASS, and Bootstrap.

Ethical and Intellectual Property Issues

Ethical issues for this project are mostly concerned with the security and privacy of the information being transmitted by end users over the internet. To protect this information, we are using end-to-end encryption with Public-Private keys. To further improve this, user logins are password secured and the entire server is locked down with administrative rights.

Ethical concerns regarding the use of the system can be traced to improper use of the platform to control robotic agents to perform unethical actions. Without the security of the users and the robotic agents, for example, unauthorized users could gain control over the robotic agents of another party or gain access to local information stored on the server or the robot. Another such unethical action would be to control the connected robot to perform unethical actions such as hurt another person or another person's property, to surveil another person without their permission or generally to use the telerobotics system in "bad faith" of the community it is being used in.

For intellectual property concerns, our robot and various other parts of platform will be using open source packages and tools. We need to cite them properly in our design specification according to what their licenses say. This is not only an ethical concern, but a legal concern as well. If we do not give the authors of the software we use the credit they are legally mandated to then this will be an intellectual property issue.

For our own licensing, we have chosen to go with the MIT license. "As of 2015, according to Black Duck Software and GitHub, the MIT license was the most popular free software license, with the GNU GPLv2 coming second." (Wikipedia - MIT License)

Change Log

- A major change to the proposal is that we lost our computer engineering component of our team. This was half of our members we initially had.
- The project's budget has changed drastically since our initial plan. This is due to the fact that we previously thought the computer engineering team would be providing us with a robot.
- We also now know we're going to be using the school's servers, so the expense of using AWS is no longer a concern. Removed server cost calculations from the budget section.

- Our milestones and goals for first and second semester have changed.
- We will no longer be implementing an object detection system