RoboTractor: A Secure Signaling and Control System for Remote Management of Agricultural Vehicles using XMPP and REST

Jeremy Wright
Arizona State University
jlwrigh1@asu.edu

Arun Balaji Buduru
Arizona State University
abuduru@asu.edu

David Lucero
Arizona State University
dwlucero@asu.edu

***Abstract*—The need for the automation of vehicles is increasing due to improved efficiencyrequiring improved efficiency, operation, and reduced operational costs. The scope of this project is to build a secure signaling and control system with mechanisms a focus on security, in order to drastically reduce if not eliminate the secure signal and control the systems from being are compromised, making automated vehicles vulnerable compromised. An attack against these systems could allow for the vehicles to be stolen, damaged, or otherwise harm the environmenthijacked, or cause other losses. The main tasks in this project are to build a secure front-end web interface for users to give instructions, built upon a Representational State Transfer (REST) architecture, set up an Extensible Messaging and Presence Protocol (XMPP) server and client with required authenticationand , data integrity check mechanisms and encryption, and build a simulated automated vehicle, capable of interfacing with the XMPP components and front-end.**

***Index Terms*—Secure signaling and control, remote management, agricultural vehicles automation, self-driven vehicles, TPM, root of trust, trusted platform, TrustZone, ARM, machine automation**

I. Introduction

Agriculture has a growing problem. Crop input costs such as water, fertilizer, pesticides, and fuel, are increasing. The global population is increasing driving the need for more food. Local, State, and Federal governments are increasing oversight and reporting requirements on growers. Land area available for planting, and harvesting is decreasing globally. All this pushes farmers to rely on technology to improve their efficiency while reducing recurrent costs. Take government oversight requirements [1], growers are required to track where, and how much material is spread into the fields. This can be a considerable overhead for growers, however an automated machine can do this quite easily, and in real-time if desired. This project demonstrates how to securely connect an agricultural machine to an automation server in a secure manner using existing internet technologies. Using existing technologies is a critical point. The market is already stretched thin, and incurring new infrastructure in rural areas can be prohibitively expensive, however leveraging the existing Internet infrastructure we can provide high levels of service in a secure manner.

The most important requirement in developing a secure signal and control system is ensuring authentication of vehicles and data integrity, here the and securing communication channels. The path plan and location feedback transmission needs to be secured and protected from eavesdroppers, preventing attackers from compromising and/or stealing these vehicles. In this project we use Django the Django web framework to handle the front end interface for the users. We use python libraries and server duties. Django facilitates the use of python libraries and components to interface between user and path generator serverwhere we use XMPP and REST protocols, and XMPP clients to control the agricultural vehicles. We expect our Front-end, server, and client-side vehicle simulators. The end-goal of RoboTractor is to have a comprehensive tool that translates the abstract user inputs into remotely executable actions for agricultural vehicles in a secure manner . using advanced web, network, and cryptographic techniques.

RoboTractor is a framework to provide integrity,

authentication, and root of trust services for machine automation specifically within an agriculture domain. Electronic Control Unit (ECU) manufacturers will particularly be interested in the root-of-trust details described in Section III-B. Agricultural service providers have typically be subject to locality issues. Historically, it's been difficult to provide agricultural services at a reasonable cost, at any appreciable distance from the farm itself. RoboTractor provides a conduit for service providers to provide their services at a distance. This allows expert agronomists to consult with farms at any distance. Not only can this reduce cost, but it also allows growers access to a marketplace of services before unavailable to them. RoboTractor is the foundation for connecting the needs of the growers to the providers.

II. System Models

A. System Model

RoboTractor will leverage the Django Web framework [2] , to realize the required interfaces. Figure 1 describes the connection of these components. The combination of XMPP and REST in RoboTractor is a demonstration of how to extend the existing HTTP development environment i.e. "the web of things" into a stateful protocol. HTTP is by design stateless. XMPP on the other hand is a stateful streaming connection between two clients. In this case the Tractor and a command server. To achieve this mesh we will leverage the Bidirectional-streams Over Synchronous HTTP standard protocol [3]. BOSH provides a standard mechanism to operate the streaming XMPP protocol efficiently over an HTTP connection. This is essential for a scalable webservice.

B. Software

1) XMPP Server: RoboTractor uses the ejabberd [4] XMPP server as it provides an existing Python interface, to integrate with the rest of our Python based ecosystem.

2) REST Interface: TastyPie provides REST [5] by extending the existing Django Models.

3) BOSH Interchange: Punjab is an Django plugin implementation of BOSH [6]. In addition to BOSH, this library combines the ejabberd Users with Django Users to provide a single authentication and authorization framework. While this demo project will have a single user type, this combination is critical to maintain proper use management and least-privilege authorization.

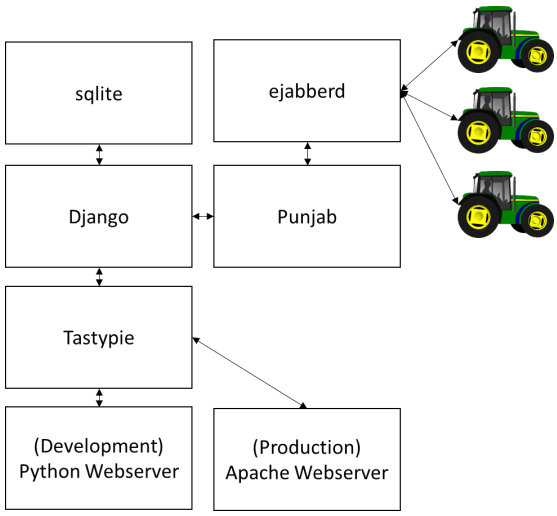


Fig. 1. Software Component Block Diagram

C. Security Model

RoboTractor deals with two primary security classes: One involving the vehicles and other assets, and one involving the system components and their communication channels. Because the project is web-based and uses PKI, numerous attack models have been created that this project will deal with. Within the class of communication channels, attacks may be present against the authentication systems of both users and assets. Robotractor ~~will employ~~ currently employs advanced two-factor authentication. Usage of TLS and other best common practices will ~~help~~ be implemented to prevent against replay attacks on the system, and also for establishing a secure session between assets and control. A dual-encryption scheme has been devised to send encrypted control signals within an encrypted XMPP stanza to ensure high information assurance. Because this project will utilize PKI, key protection is important. To this end the RoboTractor project plans to secure keys on assets with a Trusted Platform Module or relevant simulated facsimile. By making both key-pairs private and secure, RoboTractor provides advanced communication security. On the other end, the server components must be secured

as well using best common practices in server defense, firewall, patches, updates, etc. Additionally, the RoboTractor system will implement an advanced GPS location reporting system to provide physical security in case of asset theft. By defending against these attack models, the RoboTractor project presents the realization of an advanced system for signaling and control with high security.

III. Project Description

Implementation is divided into 2 primary phases (Marked as milestones on the attached Gantt chart). The first phase is an integration phase connecting the various libraries and off the shelf components. It is critical that this integration does not weaken, but rather strengthens the security properties provided by each independent component.

The second phase is the extension of these components to provide command and control of agricultural vehicles. For schedule mitigation, David and Arun are scheduled to begin this phase before the framework is assembled. Once the framework is in place, David and Arun's interface designs may be merged with the overall system.

A. ~~*Task 1 : Development Environment*~~

~~The development environment consists of configuring all the tools needed to portably work with the software package. This will involve installing project dependencies, and deployment scripts to allow all members of the team to work effectively together. The complete environment will be stored in git.~~ As of the creation of this document, all Midterm deliverables have been achieved. The following tasks have been updated to reflect work currently done, and remaining future work.

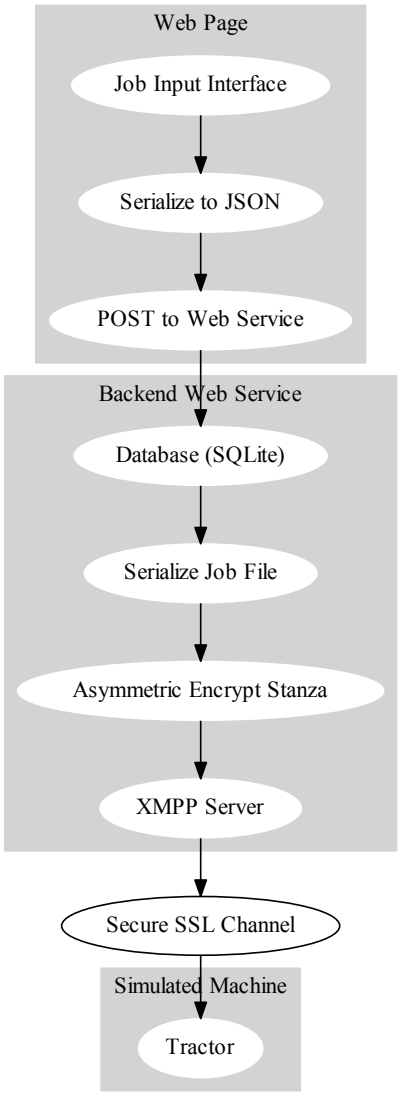


Fig. 2. High Level Signal Path

A. *Task 1 : Development Environment*

This task consisted of setting up the development environment for the RoboTractor project. A project homepage was provided to us via the ASU virtual lab project. This home page provides source control in the form of a GIT repository which our project takes advantage of. Additionally, two virtual machines were provided via the ASU virtual lab project to use as our project servers. Setup of these servers was automated with the creation of a script in order to download all required software packages. Remaining work for this task is to enable proper connectivity between the two assigned Virtual Machines.

RoboTractor on built on the Django. The development environment is best described in [2]. The Development environment consists of 2 Fabric configurations one for development and a second for deployment. This allows development settings to be more relaxed facilitating quick coding cycles, while

the deployed configuration is more secure. Separate database also prevent test data from polluting the deployed demo project.

B. Task 2: XMPP Implementation

XMPP Implementation is a configuration task to setup the ejabberd server and connect it into the HTTP Framework. ~~Once this is in place the XMPP Clients may start working.~~ The XMPP client/server components and interfacing with them is not a midterm deliverable, and is to be finished before final submission.

XMPP provides an SSL/TLS connection between the tractor, and the server. This allows a base level of secure communication. However to augment the security model, we provide additional asymmetric encryption of the command structures in each XMPP stanza. This also leverages a Trusted Platform Module (TMP). TMP is a hardware module, either part of the compute engine complex or as in ARM processors integrated into the CPU core itself. ARM's offering is called TrustZone. These security services provide a hardened storage location for private keys. One cannot extract the key without destroying the device, and the key itself. These devices also leverage a best practice called "Root of Trust" to protect against known cipher attacks [7]. The hardware modules only decrypt/encrypt material if the system is in a "secure" state. Typically this is obtained by the compute core booting an internal, untamperable ROM loader. This loaders verifies the second stage bootloader with a cryptographic signature stored in the TPM device at production time. If this signature passes, the TPM is unlocked and the hardware may continue booting. If any signature fails, the TPM remains locked, and refused to cipher information. For the purposes of RoboTractor this behavior will be simulated at the API level. However hardware integrators may leverage the best practices laid out here to implement a "root of trust" to include the automation server.

C. Task 3: Demo Tractor

~~The Demo tractor is the complementary component to the Frontend UI. This is the virtual machine, a piece of software which simulates a real tractor , or agricultural vehicle~~

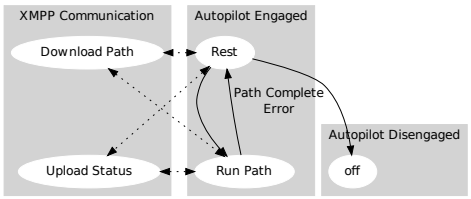


Fig. 3. Tractor Software State Machine

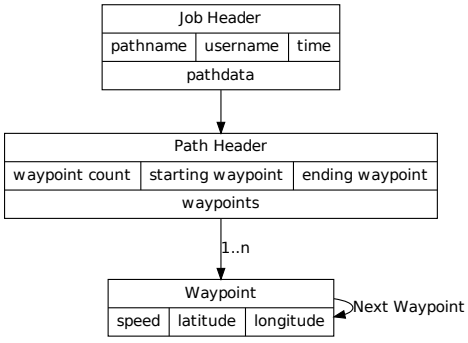


Fig. 4. Job File Format

The midterm deliverable for this task was to finalize the design specifications of the tractor simulator. This has been achieved after multiple team discussions, and a basic state machine vehicle simulator has been written in python. As more tasks are completed, the simulator is to be updated with its required functionality such as XMPP communication, path processing, and status reporting. The state machine consumes the Job File format described in Figure 4.

As described in Figure 2, the job files are stored in the database. When the Tractor is instructed to run a job, the Tractor downloads the Job file from the server over its XMPP connection. The backend server is responsible for translating the database models into JSON encoded Job File [8].

D. Task 4: Frontend UI

The "single-page" web application is the modern design methodology to web ~~apps~~ applications today. Leveraging this design architecture the Frontend will query the REST API to draw the position of all tractors within a Google Map context. Additionally,

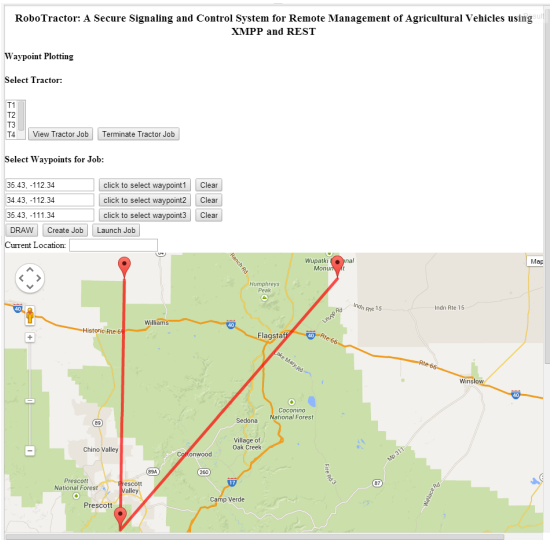


Fig. 5. Job Input Tool

the frontend user interface is to be highly secure. This task has 2 high level sub tasks:

1) Path generation
2) Map rendering
3) Two-Factor Authentication

Path generation is the primary deliverable of this ~~project~~task. The user shall be able to input a path for a given tractor to drive. ~~The Tractor~~ Waypoints input by the user, and sent to the backend web-service via the REST interface. The webservice stores the waypoints in a Model format until instructed to convert it to the Job Format and send it to the Tractor.

The Tractor will then receive this path over the ~~REST to XMPP gateway~~XMPP channel. The UI ~~may then periodically query the~~ periodically queries REST endpoint for the current position of the tractor and ~~update~~ updates it on the Google Map. Currently, a test-interface has been completed to meet the midterm deliverables, with additional functionality forthcoming.

E. Task 5: Server Backend

The Server Backend is the critical component which ~~plumbs all the componentstogether~~connects all components. Extensive knowledge of Django will make this task easier. As it will require linking multiple components together in a orchestrated fashion. We have currently setup and tested our server running on Django, and will continue to update as needed over the course of the project.

This task also includes the Service Side Session-Based Authentication. The Server is responsible for generating the Time based One-Time Password (TOTP) as described in RFC 6238 [9]. TOTP provides the "thing you have" authentication factor. RFC 6238 describes generating a QR code from a private seed value. This setups up the preshared key between the user and the web service. At this point the user may use an standard authenticator app, such as Microsoft Authenticator or Google Authenticator. At each sign on, the user is prompted for their password "the thing you know", as well as a random number from their device, "the thing you have". Additionally, this helps prevent replay sign-on attacks since the transaction requires knowledge of the preshared key to generate the unique sign on credentials. The midterm presentation provides a video demo of this functionality.

F. Task 6: REST Interface

The REST interface will server the primary means of interacting with the site. The Web interface will exist as a "single-page" app ~~who~~ which leverages AJAX principles over this REST API. The REST interface is currently functional, and will be updated to interface further with the frontend interface and backend components as they are completed.

The REST API is self documenting in the JSON format. Each endpoint publishes the data, as well as its supported services on the `/schema?format=json` endpoints. While this provides API developers a straightforward way to view the supported data, and required formats, being in JSON, a developer can also dynamically discover the API and generate a site against it. This is a common idiom in enterprise web applications using dependency injection. RoboTractor supports this idiom over REST for more advanced use cases.

Lastly, since the REST interface is a command interface to actual machines, it is critical that all requests are authenticated. The REST interface requires all queries to come from two-factor authenticated sessions (as managed by the backend

service). Each transaction contains a secured Cross-Site Forgery Request Token, and all transport is secured by SSL. This is in line with best practices for web services [10].

G. Project Task Allocation

From the tasks outlined in ~~section III~~Section III, the breakdown of work ~~will be~~ is as follows: Jeremy Wright ~~will work on configuring~~ has configured the development environments and the REST Interface. ~~He will operate~~ Jeremy is also looking into modelling trust zones which would allow us to simulate trusted hardware in actual vehicles. He is operating as the project lead due to his experience in Python-based Web Applications. Arun Balaji Buduru ~~will work~~ is working on the Server Backend and Frontend UI components. ~~David Lucero will work~~ He has completed a front-end test interface and is continuing to update it to achieve all required functionality. David Lucero is working primarily on the ~~Demo Tractor~~ Tractor simulation agents and also XMPP implementation. A barebones tractor simulation has been completed and will be continuously updated as the project progresses. Each group member ~~will have~~ has approximately 30% workload of the project with the remaining 10% shared due to the interconnected nature of all components. A detailed task breakdown can be seen in the attached Timeline.

H. Deliverables

Midterm Deliverables:

1) Functioning REST Interface (completed)
2) Front-end Test Interface (completed, with additional two factor-authentication)
3) API documentation for using the REST interface as an external service. (completed)
4) API documentation for using the XMPP interface to act as a vehicle. (moved to final deliverable)
5) Finalized Design of Vehicle simulator (completed, with additional python test implementation)
6) Interim Progress Report (completed)
7) Updated Project Schedule (completed)
8) Interim Project PowerPoint Presentation (completed)

Final Deliverables:

1) Finalized back-end ~~configuration~~ server configuration including easy installation script(s)
2) ~~Administration panel for configuring vehicles. Finalized~~ Completed, secure front-end interface including two-factor auth and SSL/TLS (complete)
3) Vehicle Simulator with working component interfaces and simulated trust zone functionality
4) Final Project Report
5) Final Project PowerPoint Presentation
6) User-guide covering front-end interface, and including API documentation for REST and XMPP interfaces

I. Project Timeline

The attached project timeline (generated from Microsoft Project) describes the overall tasks of the project.

IV. Risk Management of the Project

Several potential issues have been identified that may pose risk to successful completion of this project. These risks have been identified in Table I. Along with the description, ratings have been assigned to each risk identified, and potential mitigation strategies are described.

V. Conclusion

~~In this proposal we intend to build a remote control system leveraging existing internet technologies XMPP and REST . Future work based on this approach may include asset managementfor corporate farm management.~~ This work continues to explore how REST and XMPP can help alleviate the growing cost issues in Agriculture. By providing a base set of secure services future integrators can extend the provided data services over this base framework. Using the RoboTractor framework accelerates development by allowing content providers to provide secure, integrity, and authentication to their own agricultural focused service.

Agricultural machines themselves are growing in complexity, and the number of ECUs, integrated sensors, advanced engine management, government oversight documentation, prescription maps,

real-time agronomy, crop consulting, genetic seeding, the list is unbounded. These services can all be accelerated, reduced cost, and increased quality and timeliness with the addition of secure connectivity. RoboTractor provides the framework for these content provides to provide such services. It's an exciting time to see where growers are incorporating technology to bring us our daily bread.

References

[1] (). Growers now have fertilizer, diesel cost information available | cotton content from southeast farm press, [Online]. Available: http://southeastfarmpress.com/cotton/north-carolina-report-will-track-fertilizer-diesel-costs-growers (visited on 03/08/2014).

[2] (). Django, GitHub, [Online]. Available: https://github.com/django (visited on 02/06/2014).

[3] I. Paterson, D. Smith, P. Saint-Andre, and J. Moffitt. (Jul. 2, 2010). Bidirectional-streams over synchronous HTTP (BOSH), [Online]. Available: http://xmpp.org/extensions/xep-0124.html (visited on 02/06/2014).

[4] (). Ejabberd community site | the erlang Jabber/XMPP daemon, [Online]. Available: http://www.ejabberd.im/ (visited on 02/06/2014).

[5] (). Toastdriven/django-tastypie, GitHub, [Online]. Available: https://github.com/toastdriven/django-tastypie (visited on 02/06/2014).

[6] (). Twonds/punjab, GitHub, [Online]. Available: https://github.com/twonds/punjab (visited on 02/06/2014).

[7] (Jun. 2013). Implementing hardware roots of trust, [Online]. Available: http://www.trustedcomputinggroup.org/files/temp/76882F9C-1A4B-B294-D09D38B918AD23D0/SANS%20Implementing%20Hardware%20Roots%20of%20Trust.pdf (visited on 03/09/2014).

[8] (). JSON, [Online]. Available: http://www.json.org/ (visited on 03/09/2014).

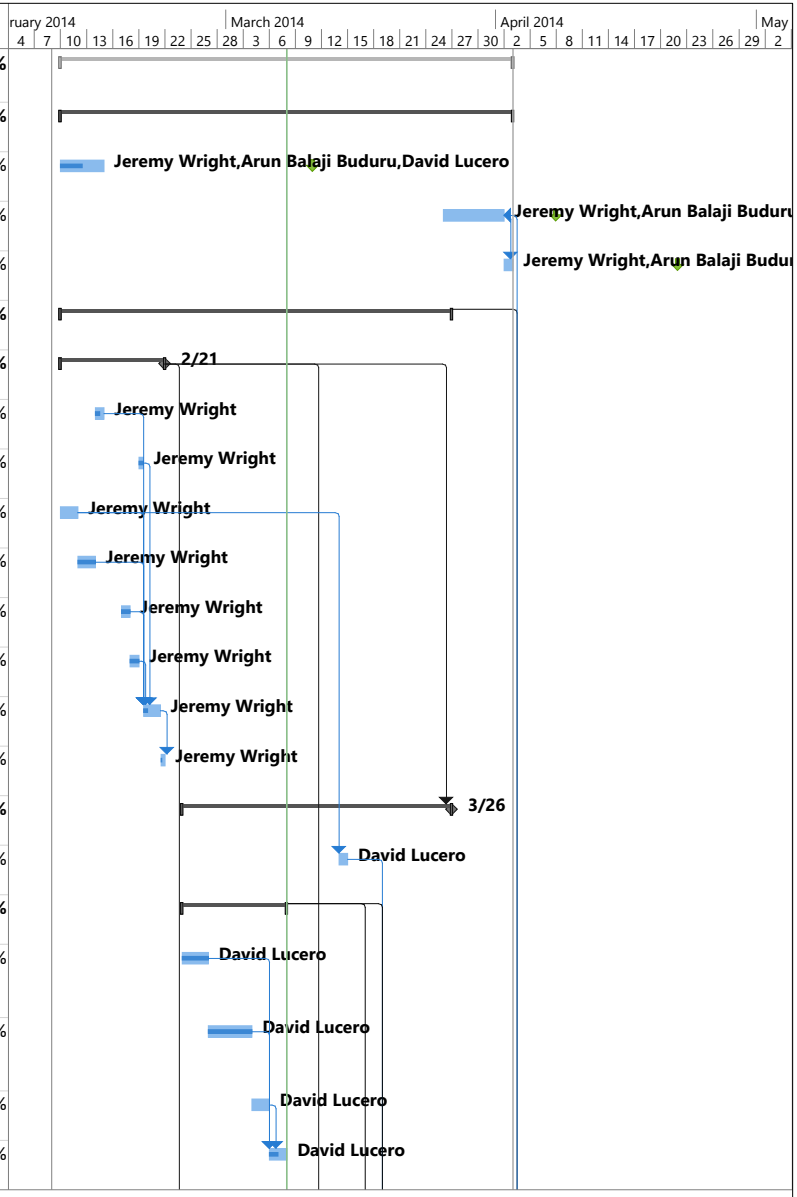
[9] J. Rydell, M. Pei, and S. Machani. (). TOTP: time-based one-time password algorithm, [Online]. Available: http://tools.ietf.org/html/rfc6238 (visited on 03/09/2014).

[10] (Oct. 1, 2002). Best practices for web services: part 1, back to the basics, [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-best1/index.html (visited on 03/09/2014).

TABLE I
RISK MANAGEMENT

Risk Description	Risk of Failure	Consequence of Failure	Mitigation Strategy
Connections between components must be secure	Low	Product would still function, but would lack security information assurance will suffer	Plan to include security on component connections ahead of time. Follow best practices (SSL/TLS)
Evaluation of Project depends on proper vehicle agents	Medium	System is unable to be tested if vehicle agents are inoperable or incorrect	Vehicle agents will be designed first to ensure compatibility with all other components. Thorough review and testing of component to ensure proper functionality
Improper configuration of components	Medium	Components do not communicate properly with each other. System functionality is void	Thorough reviews and testing of components to ensure proper functionality and communication
Incorrect PKI implementation and/or configuration	Medium	Puts all identified security domains at risk. PKI used becomes useless. System becomes vulnerable to outside attacks i.e. MITM	Follow best practices and standards. No re-invention of the wheel
Service Uptime (web reliably)	High	Product is unusable if network connections are not operable	Redundancy. Cloud hosting to alleviate hardware reliance

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	% Work Complete
0		**RoboTractor XMPP and REST Command and Control Server**	**38 days**	**Mon 2/10/14**	**Wed 4/2/14**			**42%**
1		**Administrativa**	**38 days**	**Mon 2/10/14**	**Wed 4/2/14**			**23%**
2		Interim Report	1 wk	Mon 2/10/14	Fri 2/14/14		Jeremy Wright,Ar	50%
3		Final Project Report	1 wk	Wed 3/26/14	Tue 4/1/14	38FF	Jeremy Wright,Arun	0%
4		Project Demo	1 day	Tue 4/1/14	Wed 4/2/14	3	Jeremy Wright,Ar	0%
5		**Implementation**	**33 days**	**Mon 2/10/14**	**Wed 3/26/14**			**54%**
6		**Development Environment**	**10 days**	**Mon 2/10/14**	**Fri 2/21/14**			**58%**
7		Fabic Deployment	1 day	Fri 2/14/14	Fri 2/14/14		Jeremy Wright	50%
8	✓	Django Project	0.5 days	Wed 2/19/14	Wed 2/19/14		Jeremy Wright	100%
9		ejabberd Configuration	2 days	Mon 2/10/14	Tue 2/11/14		Jeremy Wright	0%
10	✓	Web Server Configuration	2 days	Wed 2/12/14	Thu 2/13/14		Jeremy Wright	100%
11	✓	TastyPie Plugin Configuration	1 day	Mon 2/17/14	Mon 2/17/14		Jeremy Wright	100%
12	✓	Python Virtual Environments	1 day	Tue 2/18/14	Tue 2/18/14		Jeremy Wright	100%
13		Testing	2 days	Wed 2/19/14	Fri 2/21/14	7,8,9,10,11,12	Jeremy Wright	25%
14		Development Documentation	0.5 days	Fri 2/21/14	Fri 2/21/14	13	Jeremy Wright	50%
15		**XMPP Implementation**	**23 days**	**Mon 2/24/14**	**Wed 3/26/14**	**6**		**50%**
16		Client Login/Auth	1 day	Fri 3/14/14	Fri 3/14/14	9	David Lucero	0%
17		**Tractor Demo Software**	**10 days**	**Mon 2/24/14**	**Fri 3/7/14**			**70%**
18	✓	Command Language Grammar	3 days	Mon 2/24/14	Wed 2/26/14		David Lucero	100%
19	✓	State Machine Design	3 days	Thu 2/27/14	Mon 3/3/14		David Lucero	100%
20		XMPP Client	2 days	Tue 3/4/14	Wed 3/5/14		David Lucero	0%
21		Testing	2 days	Thu 3/6/14	Fri 3/7/14	18,19,20	David Lucero	50%



ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	% Work Complete
22		Debug Channel for Demo and Debug	2 days	Mon 3/17/14	Tue 3/18/14	17	Jeremy Wright	50%
23		Testing	6 days	Wed 3/19/14	Wed 3/26/14	22,16,17	Arun Balaji Budur	25%
24		**Frontend UI**	**31 days**	**Mon 2/10/14**	**Mon 3/24/14**			**71%**
25		Google Maps Rendering of Tractor Positions	1 wk	Mon 2/10/14	Fri 2/14/14		Arun Balaji Buduru	100%
26		Connect UI to REST Interface	2 days	Mon 3/10/14	Tue 3/11/14		Arun Balaji Buduru	75%
27		Mechanism to input a path to send to the Tractor	1 wk	Mon 2/24/14	Fri 2/28/14		Arun Balaji Buduru	50%
28		Testing	2 days	Fri 3/21/14	Mon 3/24/14	25,26,27	Arun Balaji Budur	50%
29		**Server Backend**	**6 days**	**Wed 3/12/14**	**Wed 3/19/14**	**6**		**25%**
30		XMPP to Django (Punjab) connection	3 days	Wed 3/12/14	Fri 3/14/14		Arun Balaji Buduru	0%
31		System Models	1 day	Mon 3/17/14	Mon 3/17/14		Arun Balaji Budur	100%
32		Testing	2 days	Tue 3/18/14	Wed 3/19/14	30,31	Arun Balaji Budur	25%
33		**REST Interface**	**20 days**	**Mon 2/24/14**	**Fri 3/21/14**	**6**		**54%**
34		API Design	3 days	Mon 2/24/14	Wed 2/26/14		David Lucero,Jere	100%
35		Authentication Model	3 days	Fri 3/14/14	Tue 3/18/14		David Lucero,Jeremy	100%
36		Connection to Punjab Models	3 days	Wed 3/19/14	Fri 3/21/14		David Lucero,Jeremy	0%
37		Testing	4 days	Thu 2/27/14	Tue 3/11/14		David Lucero,Jere	25%
38		System Testing	1 wk	Wed 3/26/14	Tue 4/1/14	5FF	Arun Balaji Budur	0%