Milestone One

Florida Tech IGVC

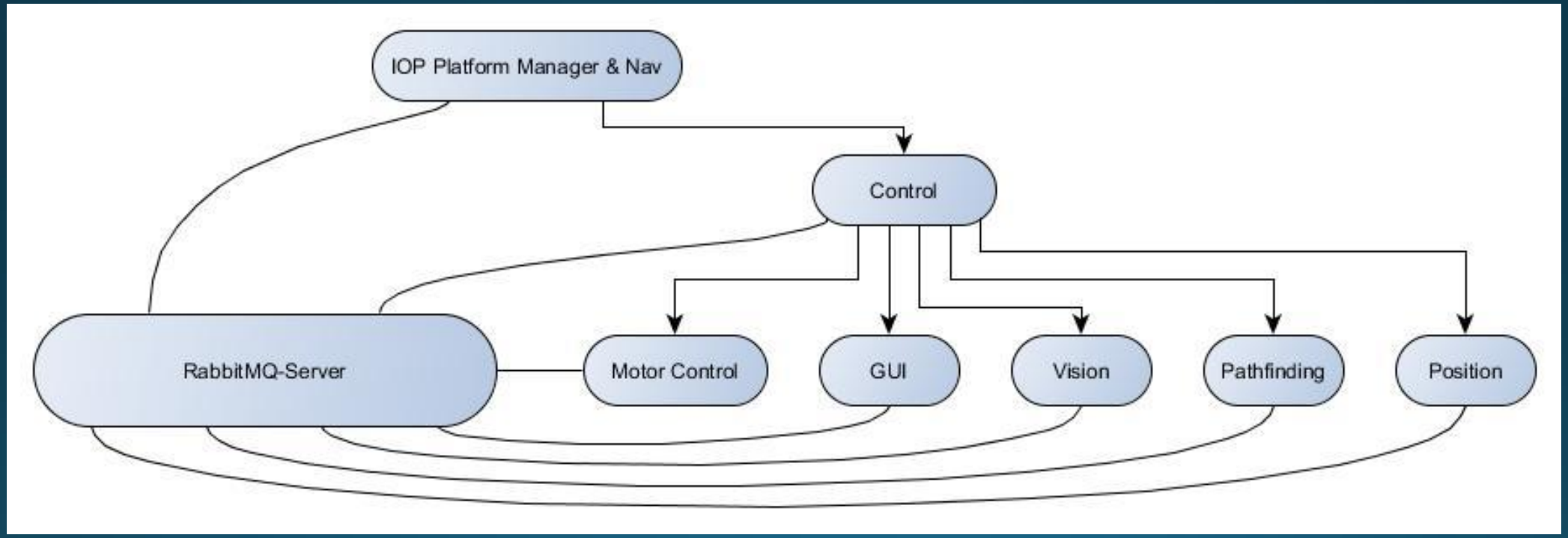
FSU Trip

- Hardware Configurations
  - Addition of LIDAR to supplement ZED stereoscopic camera
  - Routers will be added to support multiple machines
- Software Structure
  - Java & C++ will be main languages
  - RabbitMQ Server clients will support the communication framework
  - Two components shall control and log robot behavior
  - Pathfinding and motion planning shall be completed using one algorithm
  - Position finding and motor control shall be implemented by FSU
  - Computer vision shall be implemented using C++ & CUDA. The same component will also control LIDAR

Requirements Document

- Intelligent Ground Vehicle Competition (IGVC) rules constitute our customer
- Competition rules require the vehicle operate independently without human interaction other than to set up the vehicle for the course
- Software system for the robot will be comprised of independent yet interconnected modules
- Software reliability and stability is our overall key requirement

Design Document



Test Document

- Reliability is a pivotal part of the operation of the vehicle
- Isolation Testing – Each module will be tested and expected to perform by itself
- Message testing – Each module must be able to communicate using the required messaging service (RabbitMQ) to successfully communicate with the server and other modules
- Software Integration testing will be performed when each module is finished
  - Every software module will be tested with the other modules it interacts with
  - Every software module available will then be tested at the same time
- Hardware integration
  - Full testing of the entire system deployed before vehicle operation if possible
  - Vehicle operation is the final stage of complete integration testing

Pathfinding

- SBMPC Algorithm Implemented and tested using simple test cases
  - Implemented in Java
  - Underlying pathfinding algorithm: A star
  - Basic test example: placing a single obstacle between the vehicle and the goal
- Supporting classes simulate an Ackerman steered vehicle
  - Minimal code modification for application to other models
- Next steps
  - Gui interface for designing test cases, visualizing results
  - Performance analysis, optimizations

GUI

- JavaFX research done
  - Understand API, idiosyncrasies
- Internal synchronization framework developed
  - Must listen to messages, but also keep the "GUI thread" free
- Creation of basic window with all the sections as described in requirements doc
  - Ugly and barebones, no functionality
- Minimal work done
  - Pathfinding has been the priority thus far

Vision

- Completed integration of the ZED with the NVIDIA TX1 after communicating with the vendor to receive firmware updates
- Initial sets of data collected from the ZED for analysis
- Researched potential algorithms for locating obstacles and doing basic line detection
- Set up environment for CUDA, OpenCV and PCL libraries
- Implementation of object detection and line following is in progress

Communication & Simulation

- Finished simulation
- Determined best C++ client library for RabbitMQ
  (AMQP-CPP by Copernica Marketing)
- Began writing basic clients in C++
- Finished debugging Java communication client

Interoperability (JAUS)

- Standards for communication, testing, and simulation involving remote devices and autonomous vehicles
- Standards set by SAE called JAUS (Joint Architecture for Unmanned Systems)
- UDP connections with remote devices required
- Remote control/operation of the vehicle over Ethernet, local WiFi, and remote WiFi

Milestone 2 Task Matrix

Task	Will	Adam	Chris	Brent
Finished GUI	20 %	30%	20%	30%
Prototype Navigation	10%	40%	10%	40%
RabbitMQ C++ clients fully implemented	70%	10%	10%	10%
Sensor Position Estimation	20%	30%	30%	20%
Line following and basic obstacle detection	20%	-	80%	-
Prototype Control and IOP	30%	30%	10%	30%