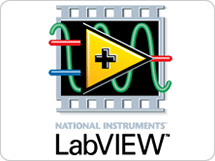
# Software Systems Design:

Using National Instruments' LabVIEW programming environment, our team developed a robotic platform implementing the basic principles of sense, think, and act. The general implementation of these principles involved performing every process of sense-think-act in parallel. Using this method is more advantageous and contrary to traditional sequential structures due to the fact that many robotic applications involve small and mobile computational architecture, relatively lengthy computation times, slow sensor refresh rates, and mission-critical safety requirements. Just like any living creatures’ behavior, the characteristics of sense-think-act are performed in parallel as well, whether it involve the subject as simplistic is a fruit fly, involving electromagnetic sensing of light, image and range recognition of obstacles, under-actuated control and in-flight stabilization. All of these amazing processes are wonders of biology are elements of controls and logic in which we would like to imitate with a form autonomous locomotion.

NI Logo

## Top level module:

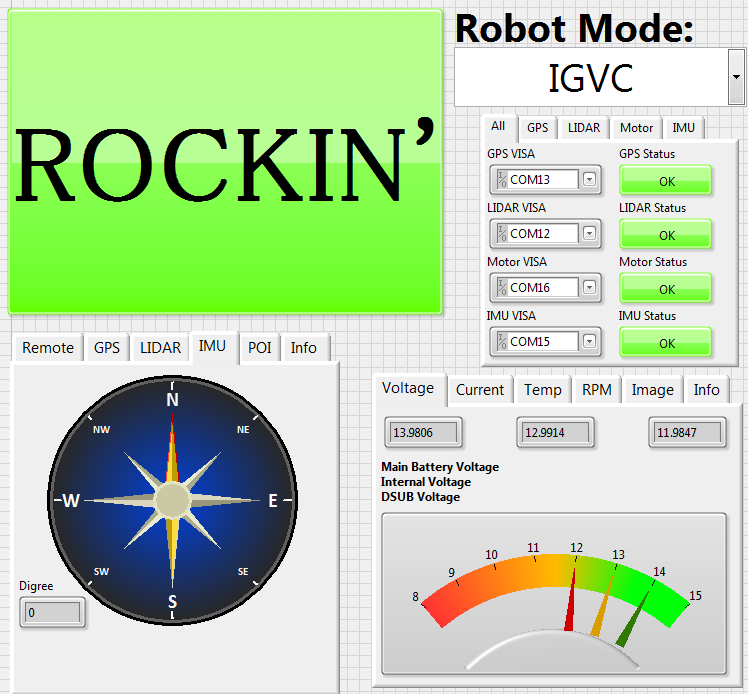
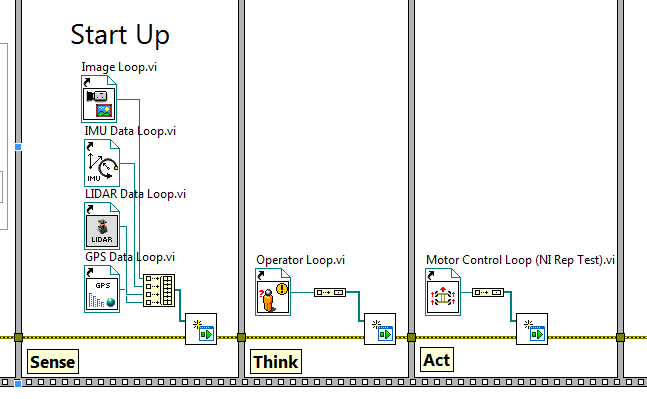
Using a top level module encapsulate the subsequent subprocesses, we're able to structure both the startup initialization, continuous operation, and shutdown procedures. Upon startup, our code first initializes all the sensor acquisition loops, which include the GPS, LIDAR, IMU, and motor telemetry acquisition loops. Every sensor loop runs independently and continuously while acquiring and writing sensor data to share global variables. This allows for different acquisition rates that are dependent on each sensor type. For instance, the GPS roughly updates once every second, while the LIDAR data updates roughly at 500 Hz. This enables for each sensor to operate at its maximum frequency, allowing for the environmental sensor database updated as periodically possible. All sensor interfaces communicate with computer platform through virtual serial RS-232 ports emulated by USB adapters. These ports are opened and configured using LabVIEW's VISA tool.

Figure : Front Panel Display

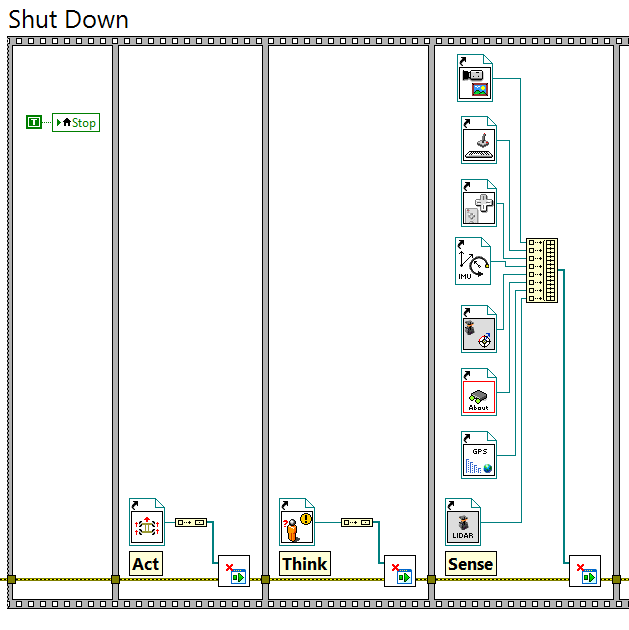
# Since:

For the LIDAR sensor loop, it continuously broadcasts polar coordinates that contain the angle magnitude of object surfaces that intersect the LIDAR's plane of sight. The GPS sensor loop acquires a longitude and latitude along with additional information such as altitude and absolute time. Using a know location approximation we can improve the startup time for the GPS by setting a close estimate of the GPS's physical location. The IMU uses a combination of accelerometers, magnetic Hall Effect sensors, and Kalman filters to continuously report its own orientation in space with respect to the Earth's magnetic field. We can then use this as a magnetic North for our compass in order to guide our robot to a command heading.



Once the entire sense network is initialized, the top level module continues to start up the logic VI, or "think" portion of the code. This VI takes it the current sensor data along with the current set mode that defines the robot's behavior, which include a “tele-op” mode for remote operation and drive control with external Wii mote or any other handheld remote and an autonomous mode for intelligent course navigation.

Figure : Start up procedure initializing each loop



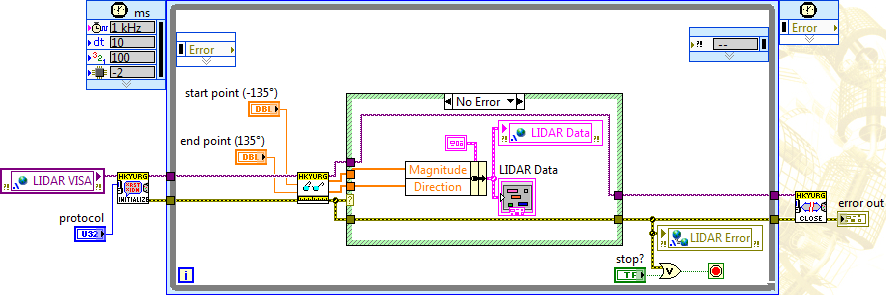


Figure : An example of our simple LIDAR acquisition loop

Figure : LIDAR, GPS, IMU, Cammra

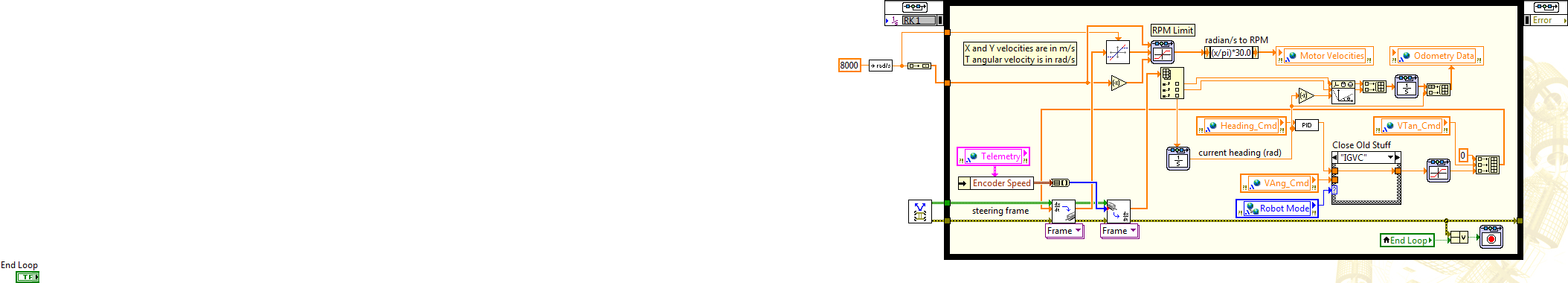
# Think:

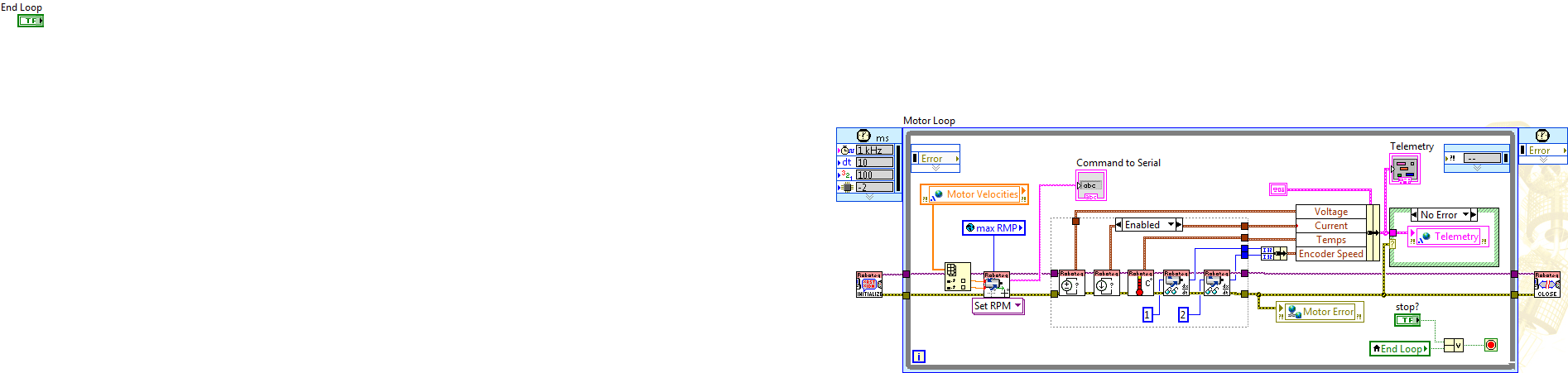
While the LIDAR data is operating in sync, the vision loop captures the white lines and stores an array of several line vectors that contain start and end points. We transpose the line vectors on top of the LIDAR distant readings to create a new obstacle histogram. Using our current GPS location and the location of the next GPS waypoint, we use spherical coordinates to calculate the current distance and directional bearing to the waypoint. Using the obstacle histogram and desired bearing, we constructed a simple bearing controlled algorithm that attempts to align the robot towards a suitable opening within the histogram field. An opening is determined “suitable” when the distance and angle of the opening produce an absolute opening wider than the width of the robot’s wheelbase plus tolerances. While searching for suitable openings the robot will also implement a cost function based on its current difference in sub-goal heading and current heading along with a current distance. If the robot veers too far from the sub-goal heading or the current distance exceeds the specified maximum tolerance, the robot will proceed to orbit around looking for alternative paths that might provide a more direct path.

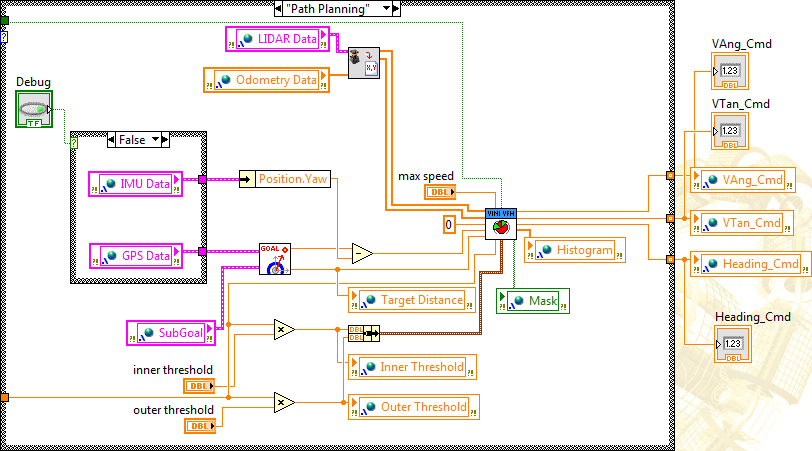


# Act:

Lastly, the motor control loops are initialized, thus implementing the "act" portion of the code. From the command bearing and command velocity specified by the logic module, the control loop attempts to drive the robot in the set direction. We calculate the specific motor velocities or RPM values by defining the drivetrain model matching our robot's own differential drivetrain. All relevant physical dimensions are specified within the model such as wheel radius, wheelbase width, and gear ratios along with clockwise and counterclockwise motor orientation. By using the system controller and implementing an integrator feedback loop, we take our command heading that serves as a set point and our current heading from the IMU that serves as the control system output. Using a basic integrator method we adjust the robots drivetrain set angular velocity to orientate the robot towards its command heading.







[Add any sensor VI’s that is in the “act” loop. Add VI’s of how the IMU acquires its bearing and sets the new bearing by taking in the global variables.]

Overall notes:

Ruffin, it’s a good start, the only problem is that you need to think about collecting and communicating your thoughts that use the principles of technical communication. Add sections to increase the understanding and communication. Add pictures that strengthen the report.

From a design perspective, start by talking very general about our software design tasks and objectives. Then start talking more specifically about our code. Mention if our final design meets the team’s goals and objectives. Also include/talk about any feature that had to be redesign after testing.

Submit a new version when done. All **red** should be removed.