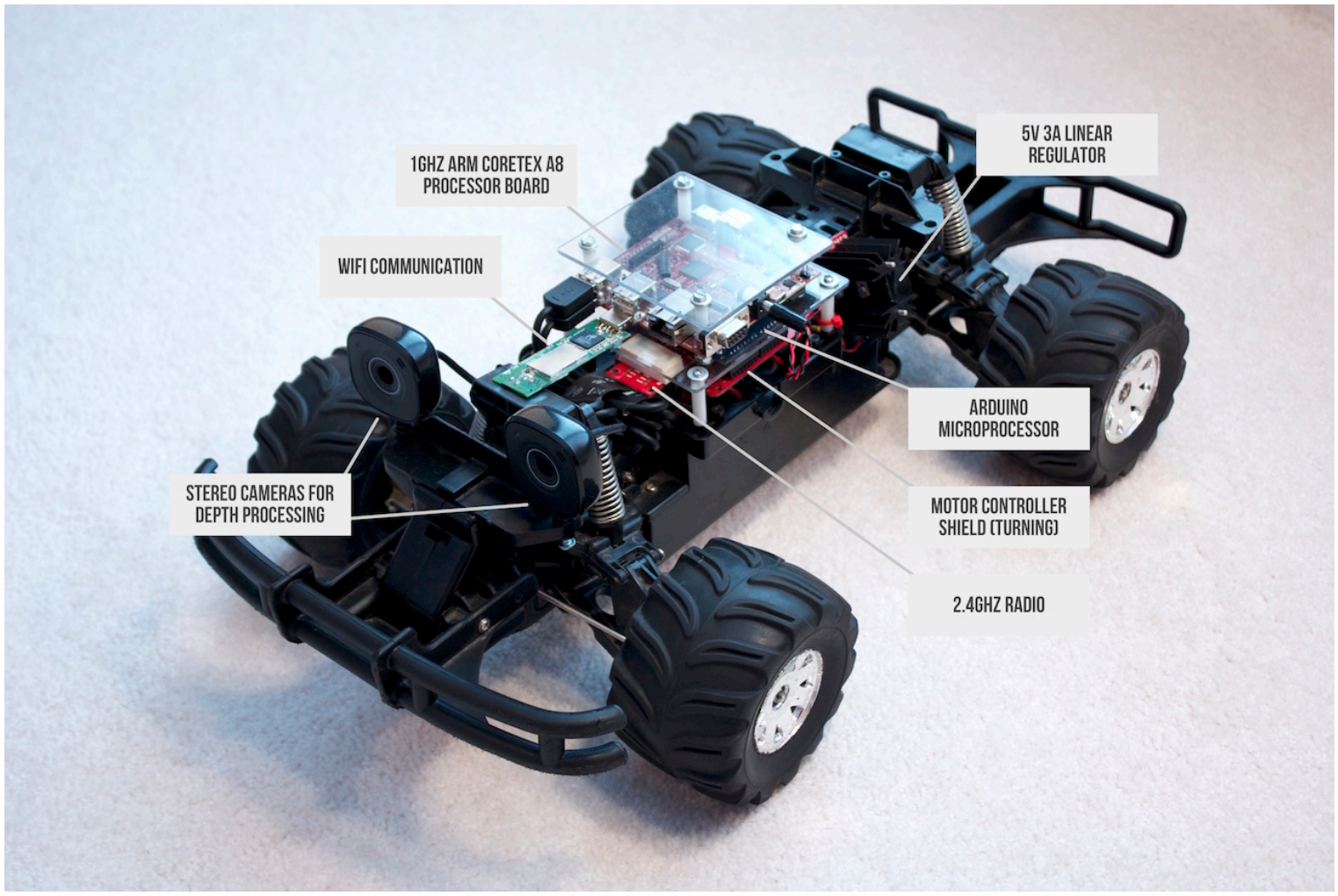
Project ARIES

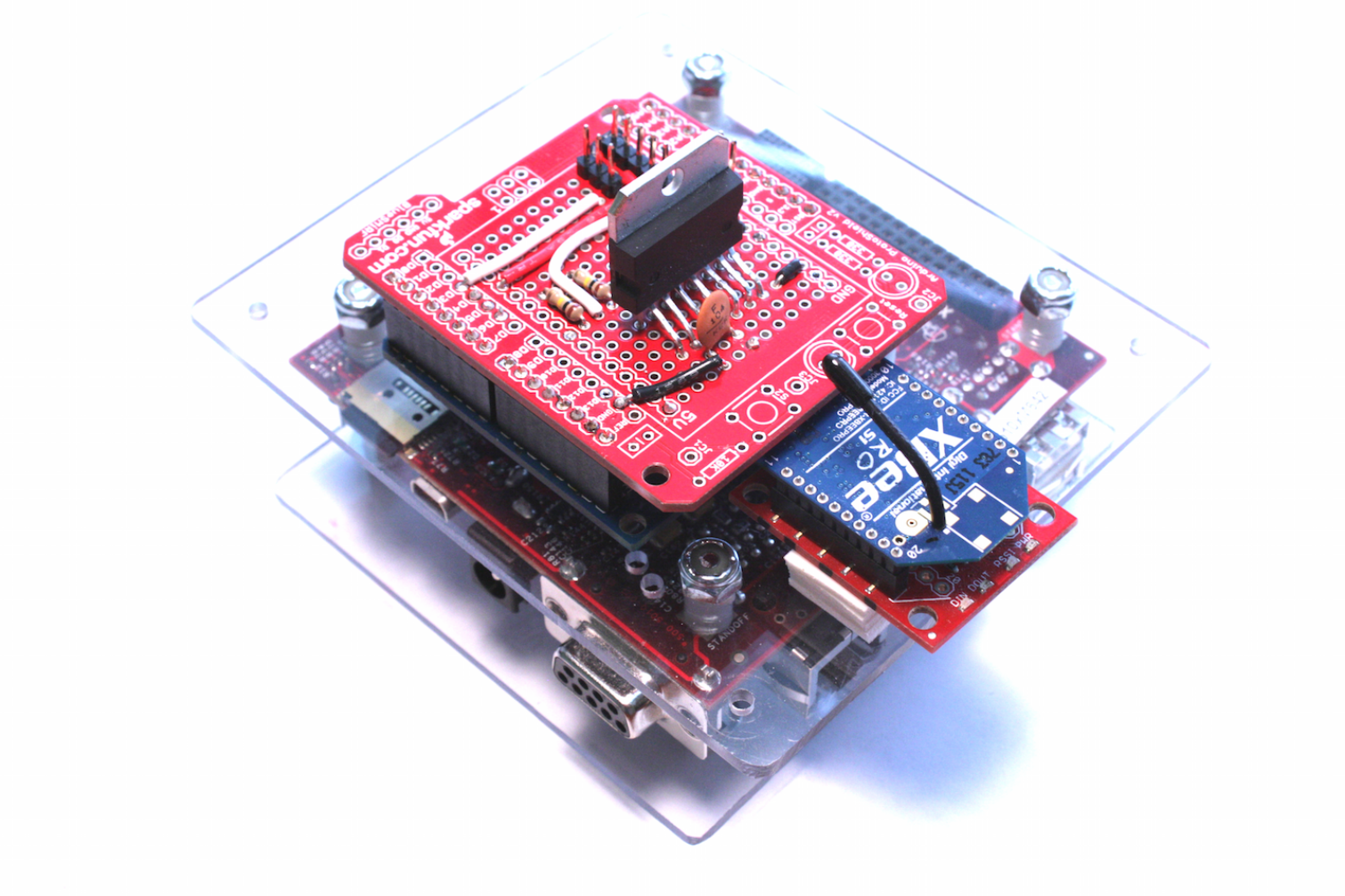


Platform Overview

Background

Project ARIES was originally designed with the intent of creating a scale car capable of autonomously and intelligently identify obstacles approaching on its course and stop or alter course before a collision occurs. I started this project as a side project the summer before my Senior year of high school and continued it throughout the year as an independent study project.

Onboard Hardware



Computing Block

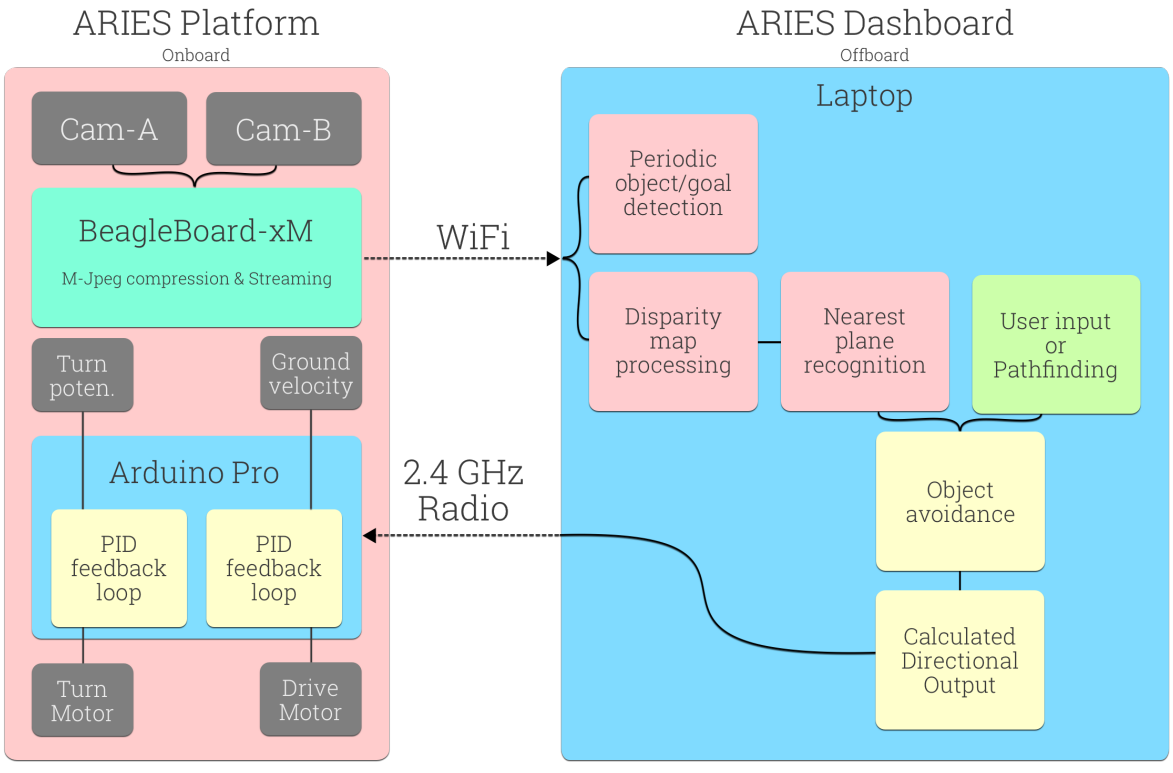
Processors

The onboard hardware consists of two main processors, the Arduino Pro board containing an ATMega328 microcontroller, and the BeagleBoard-XM containing a 1.2GHz TI OMAP3 processor. The Arduino and BeagleBoard can interface directly with each other via USB and with the dashboard via 2.4GHz radio and WiFi respectively.

Motor Controllers

The car employs two motor controllers, one for turning, and one for driving. For the turning a custom motor controller board was created based on the L298 Dual H-bridge chip that interfaced with both the turning motor and potentiometer. For driving the main motor an off the shelf 40A motor controller was used controlled via the Arduino by PWM signal.

Top Level Controls Layout



Controls Flow Diagram

On the highest level the project consists of two components, the dashboard which is run from a laptop, and the vehicle itself. All low power and control processes are run onboard either on the microcontroller (Arduino Pro) or the ARM based SBC (BeagleBoard-XM) running Arch Linux.

Onboard Processes Breakdown

BeagleBoard-XM - C++ & Bash

The onboard BeagleBoard has the sole task of transmitting the two simultaneous video streams in 320x240 resolution using *mjpg-streamer*, an open source streaming program developed for Linux. The BeagleBoard communicates sensor readings and transmits video via WiFi to a router on the same network as the dashboard. Originally the Arduino also communicated with the BeagleBoard via a standard USB port BeagleBoard, however due to software lag a separate FM radio communication channel interfacing directly with the dashboard was implemented for a reduced latency form of control.

Arduino Pro - C

The Arduino is tasked with all the low level I/O on the car, this includes sampling battery voltages, reading the turning potentiometer, and *(in the future)* reading the ground speed sensor. The Arduino also controls the two onboard motor controllers, one for turning and one for propulsion. To increase accuracy of turning a PID control loop was implemented using feedback from the turn potentiometer.

> A PID loop, for those who do not know, is a control feedback loop used to minimize accumulated error by using the three part formula composed of three characteristics; a *proportional* response characteristic, an *integrated error* characteristic, and a *derived prediction of overshoot* characteristic. By balancing the tree of these functions an aggressive and accurate output can be created. More information on PID loops can be found on Wikipedia

Offboard Processes Breakdown (Dashboard)

Dashboard - disconnected

Vision Processing

The vision processing for this project was done using OpenCV using Java. On the platform a jpeg stream is created for the left and right channel of the vision which is received by the driver station and displayed. Once the stream is obtained a stereo SGBM algorithm is applied to create a disparity map which is then also displayed on the dashboard. The vision processing was originally

designed to be done onboard, but due to limitation of low power ARM cores it was moved offboard to the dashboard instead.

Dashboard

In addition to displaying the left and right video feeds and disparity map, the dashboard also displays signal strength, battery info, enable/disable button, and PID tuning parameters for turning.

Still In Progress

As noted in the flow diagram object avoidance and ground speed sensor were not implemented due to time constraints, however they are features that may still be added.