6.111 Final Project Proposal - Paper Racing

Battushig Myanganbayar - David Gomez - Kevin Chan

1 Overview:

Our project aims to allow an autonomously controlled RC car to drive around a custom race track drawn onto a whiteboard. A user will use a heavy black marker to draw a closed loop track on a large whiteboard placed on the floor. The system we develop will analyze the drawn track and then provide commands to an RC car to drive it within the boundaries of the track. A camera pointed at the whiteboard will identify the track and store the track data. Another camera with a band pass IR filter on its lens will be used to identify the location of three IR LEDs placed on the top of an RC car on the track. This position data will be used to determine the car's location on the track and what corrective maneuvers need to be performed to keep it on the track. As a stretch goal we will allow a user to race an RC car they control against the computer controlled car.

Autonomous cars are becoming a topic of great interest in the automotive industry. This project was largely inspired by the Stanford Autonomous Racecar. Our group chose this project because it would allow us to practice using widely implemented communication protocols like I2C and more advanced peripherals, like the VGA cameras, with the FPGA. We wanted to choose a project that took advantage of the FPGA's strengths, so the live video processing and fast control algorithms involved with this project are squarely in line with that goal.

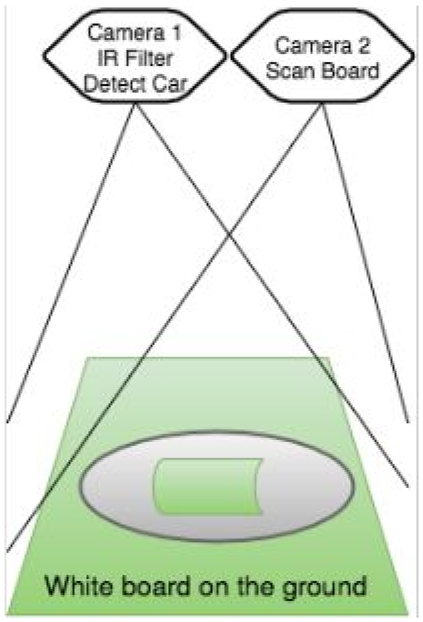

Figure 1: Physical Project Setup

2 Design:

A high level overview can be seen in Figure 2 in which we can see the 4 main modules of the project; Track Processing, Car Position Processing, Car Controller, and Video Logic.

This organization was chosen primarily as a way to allow the three members of our group to effectively work in parallel to create the most basic functionality in our project as quickly as possible. The way we have separated the modules, it is simple to test the major modules of our project in virtual testbenches using simulated inputs. This will make the integration of our modules easier. At a minimum, the user should be able to draw a track on the whiteboard and the car should follow this track. This functionality only requires the Track Processing, Car Position Processing, and Car Controller modules.

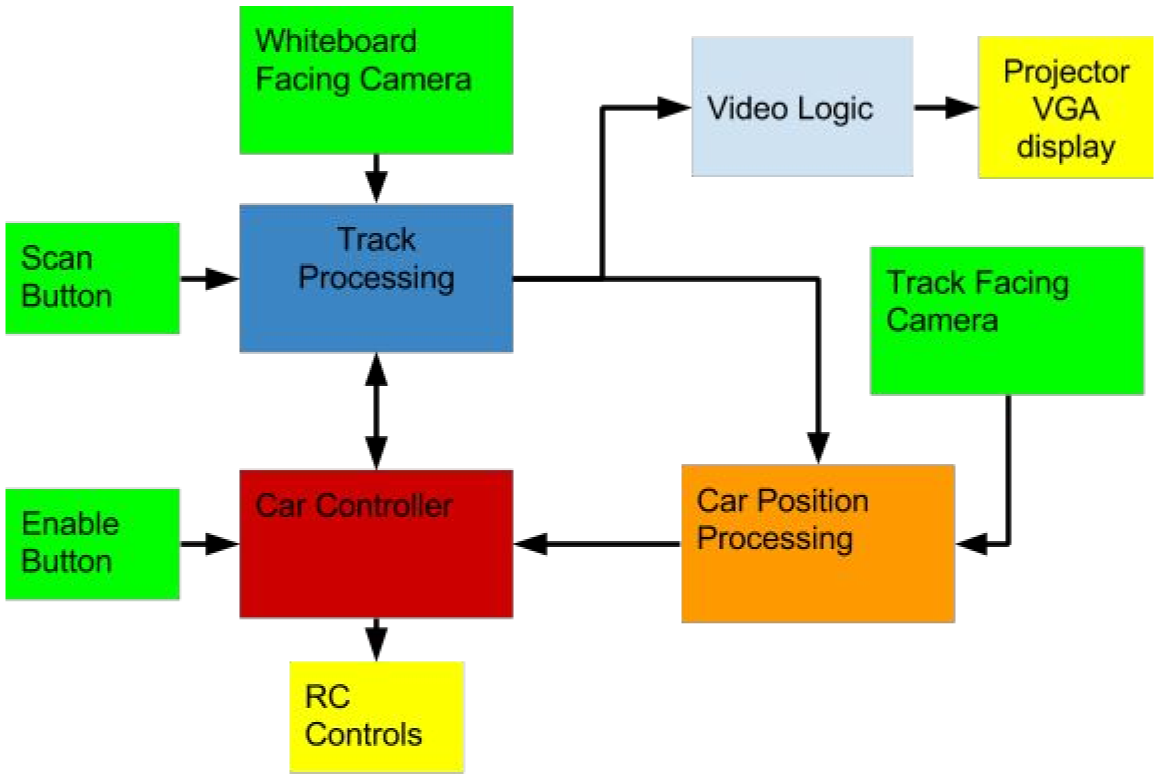

Figure 2: High Level Diagram

3 Implementation:

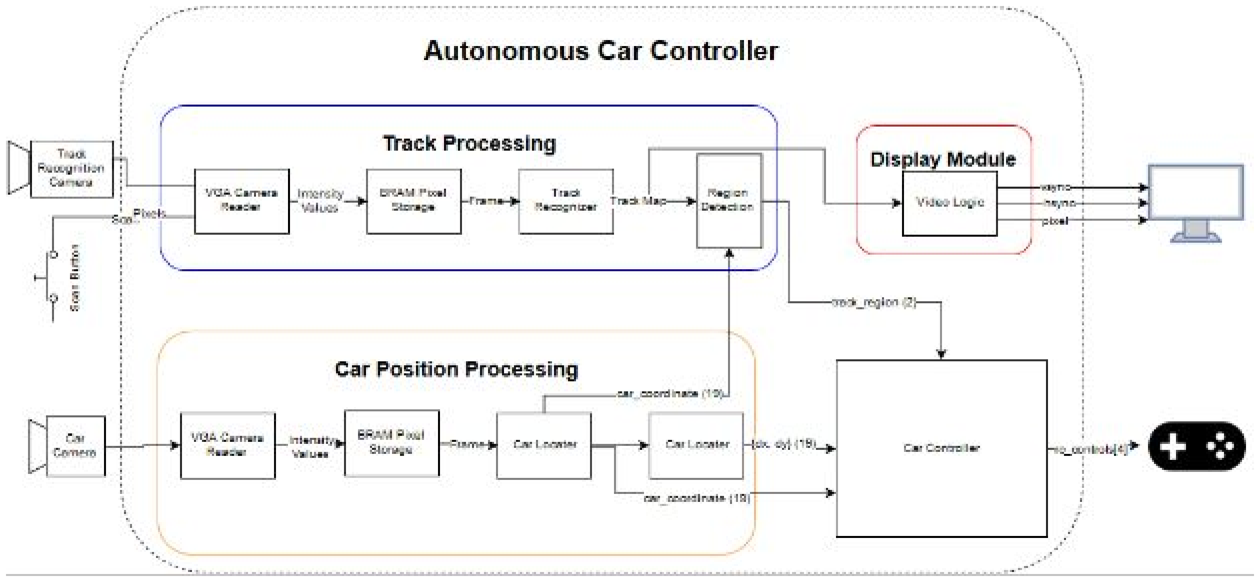

Figure 3: Detailed Block Diagram

3.1 Track Processing

The purpose of this module is to convert a hand drawn track on a large whiteboard into a representation of the track in BRAM. We capture an image of the track using the VGA camera connected to the Nexys 4 FPGA. This image is then processed to create a map of the track in the FPGA's memory. We use this map to provide feedback to our car controller module on the car's relative position to the track.

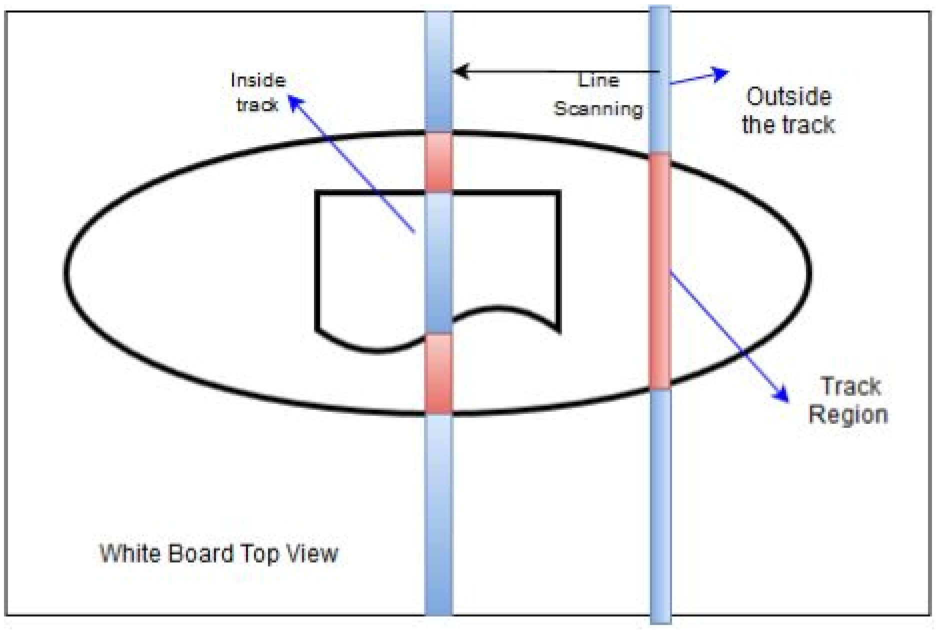

Figure 4: Interpretation of User Drawn Track

As seen above, the user draws the boundaries of the track on the whiteboard. The track recognition module is responsible for dividing the image in the FPGA's memory into the three regions as shown: the track region (the area where the car should drive), outside the track, and inside the track.

3.1.1 VGA Camera Reader and Pixel Storage

To obtain a representation of the map, we first must read in and store the pixel data from the VGA camera. The VGA's I2C output is processed by the code provided by the TA's which returns pixel values in serial. We save these pixel values in BRAM to be processed by other modules.

3.1.2 Track Recognizer and Region Detection

Once the VGA camera has stored the track image, it must be processed to determine the boundaries of the track. This will be done by implementing a line scanning algorithm, shown in figure 4. This algorithm will assign each pixel a region, either outside the track, on the track, or inside the track. These values will be assigned by starting at the top of the image and assuming the first pixels are outside the track. Pixels will then be scanned downwards, until a black line is hit. At this interface, we know we are now crossing into the track region. All white pixels seen after hitting the black line are now within the track. When we hit another black line, we know we are leaving the track region again and we now label all white pixels we see as inside the track. This region data will then be stored allowing the car camera modules to determine the region of a pixel.

3.2 Display Module
This module will be primarily useful for debugging by allowing us to display the track that we have processed on a VGA monitor. We may also have this module overlay a box representing the position of the car and it's heading for debugging as well.

3.3 Car Controller
Once the cameras have identified the position of the car and determined the regions of the track, the RC car must now be given commands to allow it to drive around the track.

This action is performed by the car controller through two main functions. The first is to take inputs from the region detection module that correspond to the current position of the car and the next predicted position of the car and determine how the car must turn to stay on the track. The second is to generate the proper command signal to send to the RC car to move in the desired way.

The turning direction calculation is done through a simple table that will store all the possible combinations of current and predicted region and then output desired turning direction that will keep the car on the track.

The command signals will be sent by utilizing the RC controller that the RC car came with. By attaching MOSFETs in parallel with the switches we are able to simulate the input of a human using the controller electronically. This is a quick easy to way to allow electronic control of the RC car; however, it has limitations caused by the RC controller's inability to process fast inputs. There may also be ways to utilize the controller's RF signal generation circuits to get finer control which we will also look into.

3.4 Car Position Processing
It is necessary to have precise information about the car's position and heading in the projected track space so that our control systems keep the car on the track. This whole module's purpose is to extract and relay any information the car controller block requires from the track facing camera. One critical detail of our implementation is that because the camera that recognizes the track and the one that searches for the car's position and heading are mounted closely together, we assume that pixels with the same coordinates in both cameras' views correspond to the same physical point on the board. This greatly simplifies the mapping of pixels from one camera to the other.

3.4.1 VGA Camera Reader and Pixel Storage

The car locater module uses the same modules as the Track Processing module to read in data from the VGA camera and store pixel values in the BRAM.

3.4.2 Car Locater

The car locater module's purpose is to report the location of the car in the camera's view. The car will have 3 IR LEDs on its roof that point upwards towards the track facing camera. Having 3 LEDs allows us to determine the heading of the car. This module will process the images from the track facing camera to determine the position of each LED in the camera's view. The module will report the "center of mass" of each LED in the camera's view as a tuple corresponding to x and y pixel indices. We report the pixel indices corresponding to the center of mass of the leading LED as the location of the car.

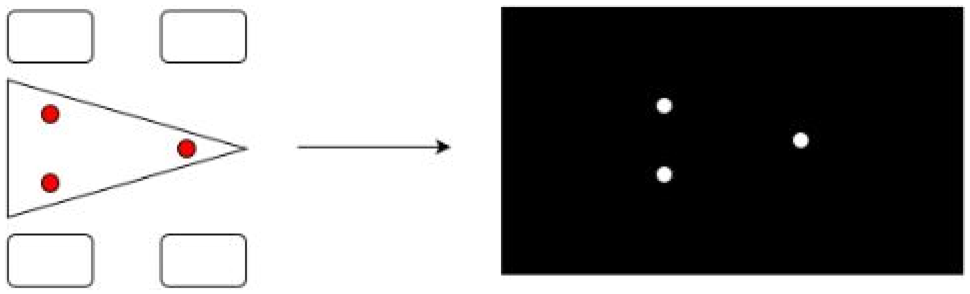

Figure 5: IR LED's on Car and image presented to camera through IR bandpass filter

3.4.3 Car Heading Identifier

We can find the heading of the car by finding the orientation of the car's LEDs relative to each other. We can arrange the three LEDs on the car's roof such that the front of the car is distinguishable from the rear. We can interpret the position of the LEDs and report to the car controller module a heading as a pair of x and y velocities.

4 Timeline:

Week/Members	11/2/2015	11/9/2015	11/16/2015	11/23/2015	11/30/2015	12/7/2015
All	Rough Proposal Draft	Project Design presentation	Revised Proposal, Project Checklist	Thanksgiving week	Buffer week, work on stretch goals	Done

Battushig	**Recognize track, generate track in memory**	**Recognize track, generate track in memory**	**Integration**	**Testing**		
David	**Get car under control from FPGA**	**Given heading and position, control loops**	**Integration**	**Testing**		
Kevin	**Identify car position, heading. Basic mapping of camera space to map space**	**Identify car position, heading. Basic mapping of camera space to map space**	**Integration**	**Testing**		

Stretch goals for this project include lap timing and counting. This will allow us to race against the autonomous RC car. To do this, we will need to place checkpoints around the track to ensure that a full lap has been driven by each car.

5 Testing:

We have divided the work for this project in a way that each module can be worked on in parallel. Each person will be responsible for developing their own ModelSim test fixtures for each of their modules. Rigorous testing on the module level will make integration much simpler. One of the challenges of testing will be reading the large memory blocks used to store camera inputs to ensure that the processed images are what we expect.

Conclusion:

Our group chose this project because we were looking for a project that involved image processing and controls- things at which FPGAs excel. We are excited to learn new ways to use the FPGAs and how to interface more complicated peripherals like cameras with them. The end goal of racing against the computer on a track we draw will be very rewarding.