EECS 373 Project Proposal - Mega Wheels

Sagar, Aaryaman	Ma, Chong	Markov, Anton	Dudkov, Artem
aary@umich.edu	machong@umich.edu	aamarkov@umich.edu	dudkoff@umich.edu

1 High Level Description

Our idea is to build a real life car battle game in which players get to control remote control cars that we buy off the market and customize so that they work in an embedded environment. These cars will be customizable before the start of the game to have targeted abilities towards either attack, defense, speed or any other factors to be determined.

The project will consist of several RC cars with SmartFusion Boards built onto them that can read in input from a controller and output to control the motion of the cars. We will refer to these as the slave boards from now on. There will also be a master "server" setup with a computer and a SmartFusion Board together. This master will be constantly reading input from the current state of the game, outputting results onto the computer and also giving the cars extra abilities and enhanced battling skills.

What sets this game apart from other RC controlled car games is that each car will have customizable powers. Before the game starts the players will be able to select what sort of car they want in software and that will determine the abilities of the car and how they progress to improve throughout the game. For example a car can be the speed car, and throughout the game as the car kills its opponents, it gains more speed, and another car can be a sharp shooter that shoots more accurately than other cars and gets better and better over time as it gets more kills. Further the car that kills a stronger car gets more points; this encourages teaming up to destroy the dominating car.

The master server will keep track of the current state of the game at all times and display the results in a nice looking UI on the computer, which can further be projected onto a screen. At the end of the game there will be a list of detailed events and top 5 kills shown on the software master node.

2 Rules of the Game

The cars will be controlled by different players. The game will compose of a time limit and the player with the maximum kills at the end of the game will win. Before the game begins the players will be able to customize their car to have different abilities. These will further be one of the many types of cars that will be highlighted in Section 5 - Software Class Hierarchy.

During the game the players will be able to deal damage to another players' car by either shooting them with the IR gun fitted on the car or by colliding with the other car. The attack points for both of these will be determined by the master node before the beginning of the game.

3 Design and Implementation

The cars will be built with identical hardware before the start of the games. All customizations will be offered to the players. So they can in essence pick the type of car they want and play the game according to that. All pre-game setup will be done on the computer master server.

Before the game begins, the players will all be able to select the type of car and indicate whether they are ready to start. This will all be given a convenient user interface on a computer.

Our choices for UI at the moment is a server built on a bootstrap frontend with HTML and CSS, the backend for which will be controlled with a web server we build. This is justified by the extensive availability and documentation available for libraries and frameworks like Flask, Django, Pyramid, Java Servlet and Gos built in libraries. We expect to make further design decisions here since TCP connections through the SmartFusion will have to be sent through some existing library that interfaces with the existing network card on the board and one that provides a programming interface for us. Or we will build this ourselves but we highly doubt that such a service will not exist. Further building our own network library will involve

further complications in the cases of latency of one connection interfering with another and this will subsequently incur the need to build a polling mechanism that will poll the network packet and preferably either sit waiting to to be interrupted.

3.1 Software and Hardware on the Cars

All cars will be fitted with SmartFusion boards and have three input sources

1. An abstracted controller that a human uses to control the car. For example, if we use a wireless Gamecube controller, the receiver for the controller will be this abstract controller.
2. IR sensor, this will be the sensor that senses another an attack from another car.
3. XBee module that gets input from a master "server" that is controlling the state of the game. So in our case the state of the game will be the powers of the car (i.e. the abilities that the car is given by virtue of its performance so far) and the current score in the game.
4. Accelerometer, this tells the car whether another car has collided with it. And based on the type of the car the amount of health varied will be different.

And they will have some output sources

1. LEDs to indicate the health of the car
2. XBee transmissions to the master
3. Motors that control the motion of the car
4. IR guns that are used to attack other cars

They are programmed to serve three purposes

1. Read in input from an abstract controller and the infrared guns. And then update the components that correspond to its parameters, for example there will be 5 leds that correspond to its health.
2. Update the components of the car to do the functions given to it by the controller. This will further interface with the infrared laser that is meant to be a gun that shoots the other cars.

3.2 Software and Hardware on the Master Server

The master "server" communicates two ways via XBee with the cars. Every time it gets an input, it computes the state of the game and sends this back to either the car that it got a request from or broadcasts back to all the cars. The master server is composed of two parts

1. A SmartFusion board that connects via XBee to the cars, and connects to a computer via TCP through an ethernet.
2. A Computer that receives the state of the game via the TCP network from the master board and displays the state in a nice UI

The master server will be keeping track of the game at all times. There are potentially two ways to do this.

1. The SmartFusion Board will receive the communication via XBee and will transmit this information to the computer via TCP through a connected ethernet cable
2. The computer itself will have a XBee receiver connected to it and will constantly poll this with the help of a device driver that comes bundled with the XBee packages.

Either way the server will be displaying the state of the game on the computer and will keep responding to the cars with response messages that can potentially contain upgrades for the cars.

4 Functional Diagram

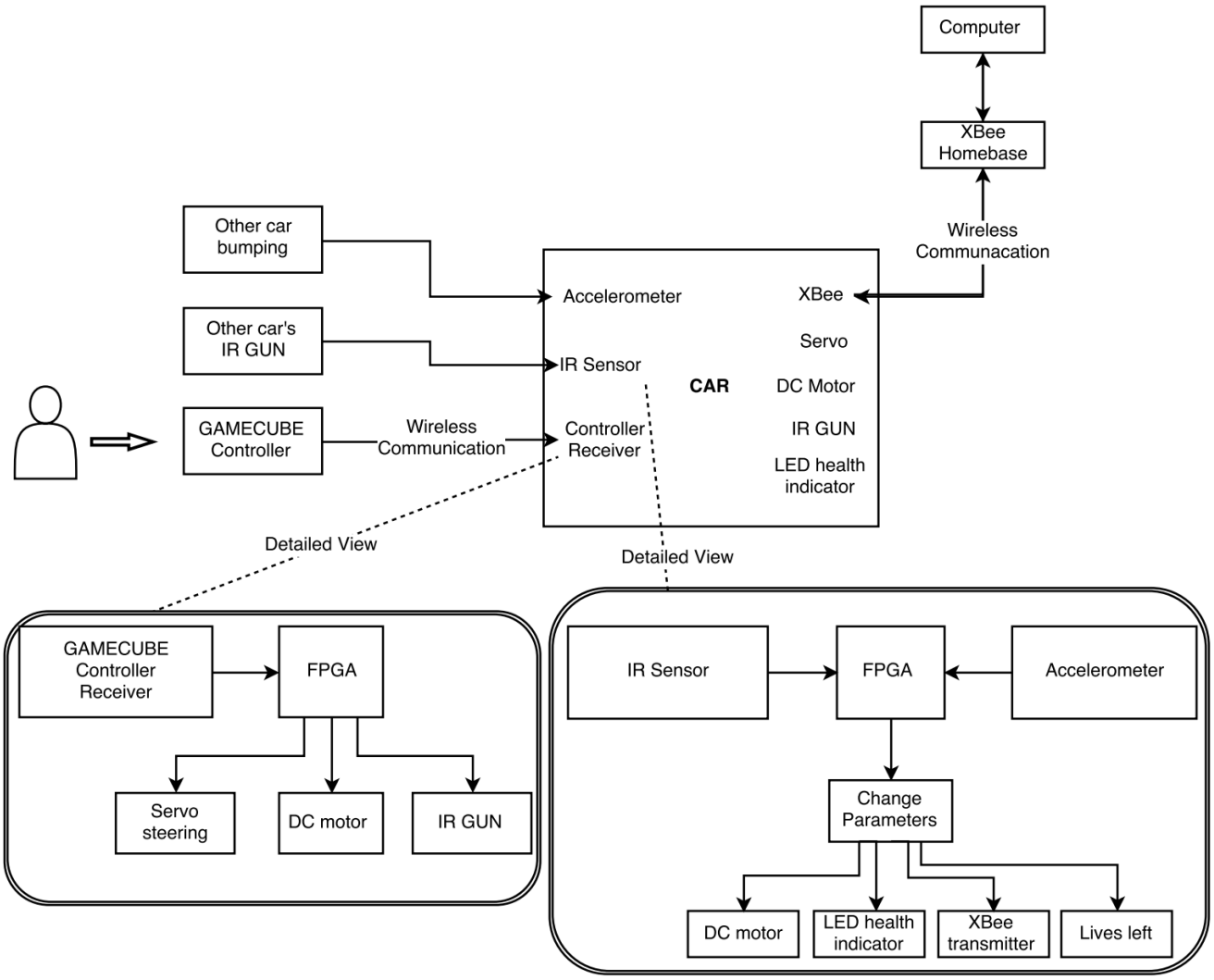

Figure 1: High level overview of the project components

5 Software Class Hierarchy

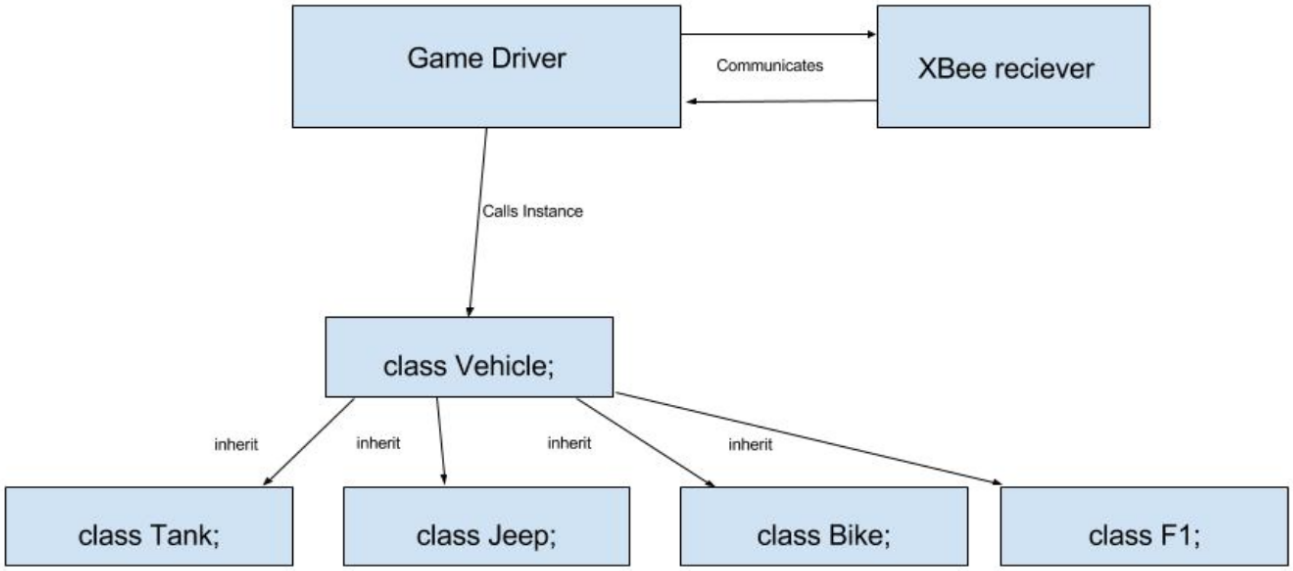

Figure 2: An Object Oriented software view of the cars

6 Component List

1. RC cars (3 - 4)
   - Driven by DC Motor
   - Servo controls steering
2. Controllers (1 per car)
   - Gamecube Nintendo 64 wireless controller and receiver
3. Computer (1)
4. SmartFusion Boards (1 per car, +1 to control main xbee)
5. LEDs (10 per car)
   - Resistors for each LED
6. 3D Accelerometer (1 per car)
7. IR transmitter (1 per car)
   - Tube to insert the transmitter into to narrow the field of view
8. IR receiver (1 per car)
   - Tube to insert the transmitter into to narrow the field of view