6.111 Proposal

Natalie Mionis and Tianye Chen

October 2016

1 Introduction

With Touch and Go, users can draw a path on a touch screen and watch as a RC toy tank follows the path in real time. We will use an Xilinx FPGA and a camera mounted on the ceiling to provide real time position and angle feedback of the car. The dimensions of the camera field of view correspond to the dimensions of the touchpad. Movements of the finger will be proportional to tank movements on the field. A simple user interface will display the path the user is drawing on the VGA display. The division of work on this project is as follows:

1.1 Natalie

1. video_decoder
2. ntsc_to_zbt
3. RGB2HSV
4. filter
5. frame_buffer

1.2 Tianye

1. touchscreen_to_vga
2. find_finger_path
3. tank_direction_and_position
4. get_direction
5. direction_generator
6. get_speed
7. speed_generator

1.3 Shared

1. Remote Control
2. Hardware Issues

2 Goals

Baseline Goals: When we provide user input via the touch screen, we want the car to roughly follow the path traced. There will likely be some delay in the tank's reaction time, so the tank won't trace the path in exact real time, but to the human eye, it should be pretty close.
Reach Goals: Time permitting, we hope to display the camera's feedback in real-time on the monitor. We would like to overlay the path drawn on the touch pad onto this video feed. As the tank covers the path drawn, the path could disappear from the screen.

2.1 Milestones

- Display the path drawn on the touch screen on the VGA monitor
- Be able to track the tank's angle and position
- Send the tank data from the touch screen
- Hardwire signals into the tank's remote controller so the car follows the path

3 Modules

1. **video_decoder**
   This module's job is to convert the input signals from the camera, outputting a 30 bit YCrCb value. This module will also output hcount and vcount to communicate the location of each pixel.

2. **ntsc_to_zbt**
   This module will take in the 30 bit YCrCb value from the video decoder, as well as hcount and vcount, to generate an address in ZBT memory. This module will then convert the 30 but YCrCb value to a 24 bit YCrCb value by truncating the two least significant bits. Then, it will convert YCrCb to RGB using a 24 bit conversion. The module will then further truncate two more bits, and store an 18 bit RGB value into the ZBT memory. The ZBT memory can store 36 bits at each address, so we will store two RGB pixels at each address.

3. **RGB2HSV**
   This module will take a 24 bit RGB value and output a 24 bit HSV value. We will pad the 18 bit RGB values stored in ZBT memory to make them 24 bits. We are choosing to use HSV because we can filter pixels based on a threshold of their hue, which will reduce noise from different lighting situations or angles.

4. **filter**
   This module takes in the 24 bit HSV values from the RGB to HSV converter as well as hcount and vcount. There will be red and green tape (or paper) on the RC car, and this module will use filters based on the hue of a pixel to either include or exclude a pixel in further calculations of the car's angle. Having filtered out the pixels below the threshold, the filter will take the center of mass of all the red and green pixels by summing their x and y values, and dividing by the number of pixels included. The module will output the two (x,y) values for the center of mass of the red and green parts of the car.

5. **frame_buffer**
   The frame buffer will store the last eight center of mass values in a circular buffer, and calculate the average of the eight values stored in the buffer. This will hopefully reduce noise in the center of mass readings since we will be averaging the last eight values, which will reduce the effect of possible outliers. The output of this module will be the average position of the car and a vector of its angle, all calculated from the (x,y) center of mass points.

6. **find_finger_path**
   The capacitive touchscreen will be the source of user input. The capacitive touchscreen communicates its x and y location through I2C. We will to decode the I2C messages to obtain the X and Y location of the finger. This module will sample the X and Y at a constant rate and store the X and Y positions in a buffer. The X and Y values will be 9 bits wide each, and so each X and Y buffer will be 18 bits long.

7. **tank_direction_and_position**
   Calculates the angle of the tank, based on the detected front and back of the car. The angle calculations will be arithmetically intensive(several squares) and will need to be pipelined and therefore delayed in clock cycles in order to meet timing constraints. The angle will be referenced to the horizontal line that intersects the tank position and be presented in degrees up to 1 decimal point. It also stores the X Y position of the tank in two registers that are 9 bits wide. The degree will be stored in a register 9 bits wide

8. **get_direction**
   Calculates an angle between the last value in the find_finger_path buffer and the current tank position, referenced to the horizontal line that intersects the tank's position. As with calculating the tank direction, calculating the finger direction will also be arithmetically intensive, and will need to be pipelined. The module compares the calculated angle with the tank angled calculated in tank_direction_and_position, and outputs a direction that is proportional to the difference. The degree will be stored in a register 9 bits wide

9. **direction_generator**
   This module will generate a signal to control the remote control so that it will move the tank to the desired direction as calculated in get_direction.

10. **get_speed**
    Calculates the speed in ft/sec at which the tank should move to the next target position by finding the euclidian distance between the last two values in the find_finger_path buffer and dividing it by the sampling rate

11. **speed_generator**
    This module will create a PWM signal to send to the remote controller of the car to control the speed of the vehicle. The input will be the desired speed of the vehicle, and the output will be the PWM signal. The speed will be calculated within the get_speed module, and will be in units of feet/second, bit shifted to the right to avoid working with decimal values. We will run tests with the RC tank to measure its maximum physical speed given the maximal speed input on the remote controller. With this measurement, we can calculate the PWM cycle time needed to drive the car at a particular speed. The PWM generator module will do this calculation and create the appropriate PWM cycle to send to the remote controller as an output.

12. **control_fsm**
    The control fsm determines the state of the tank and the physical action of the tank. The tank will first be initialized to a starting point by the user when they tap once on a certain point on the screen. Any finger inputs before the car has reached its specified starting point is ignored. When the tank has reached its initialization point, it will then be ready to receive commands from the finger and start driving. When the tank is in driving mode, the remote control receives speed commands from the speed_generator and the angle from direction_generator. When the car has reached it's desired next position, it waits for the next set of speed_generator and direction_generator commands. When the user lifts the finger from the touchpad, the tank enters a state where it waits for an initialization (single touch on the screen) to restart the process over again.

13. **touchscreen_to_vga**
    We will display that the path the finger traces in real time on a VGA Screen. The module will map the X and Y values from the output of find_finger_path onto corresponding pixels on the VGA screen. The background will be white and the path that the person traces will be displayed in an RGB color.

The mapping will maintain the touchscreen ratio of 3:4. Whatever falls out of range of the touchscreen will be displayed as black on the VGA Screen. The path that the person draws will also be thicker to accommodate for the enlarged picture displayed on the VGA screen.This module will generate XVGA signals.