6.111 Final Project: LED Pathfinder

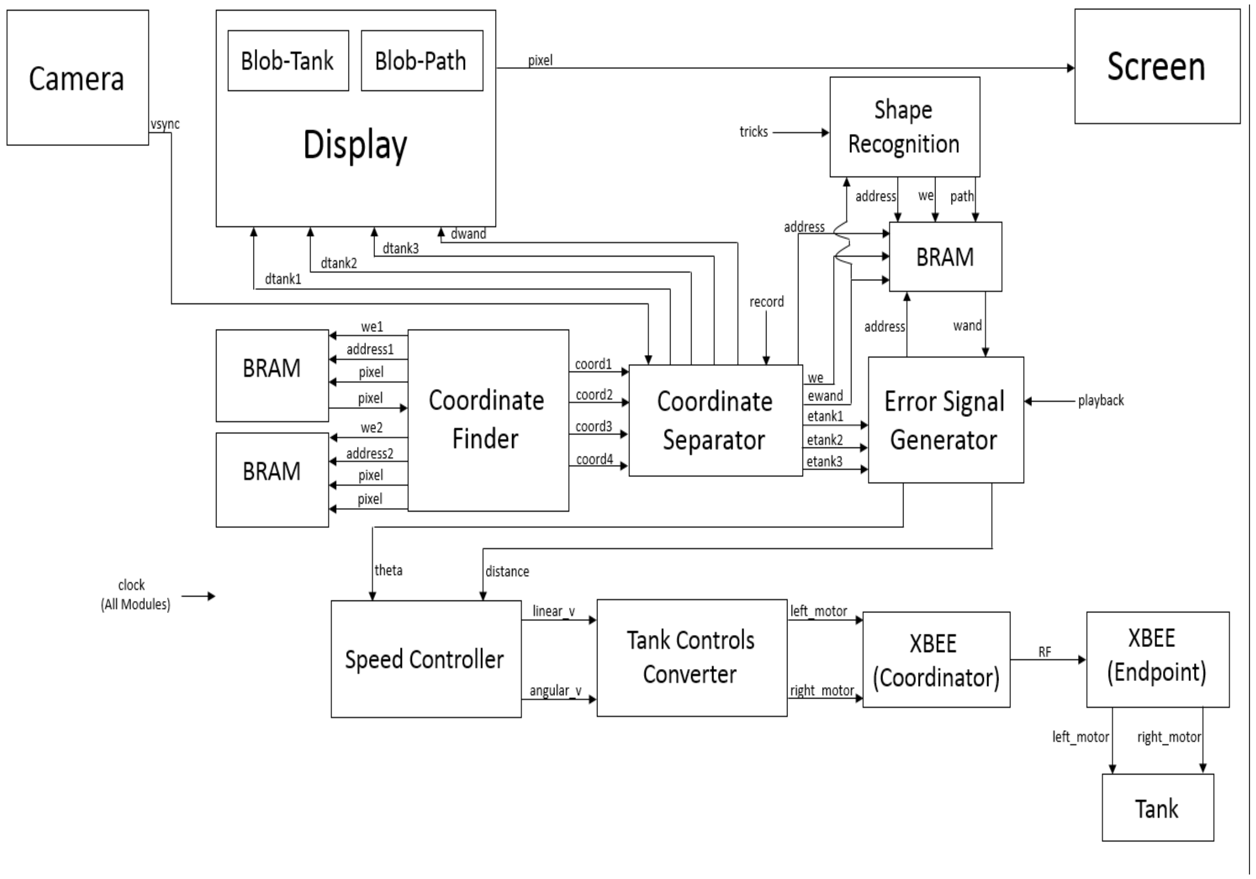
Proposal

Druck Green and Andre' Walker

1 Overview

The LED Pathfinder Project will allow a user to direct an RC tank's motion using a wand. The user waves the wand in a pattern for the tank to follow, our system stores this path, and the tank will autonomously follow the path when prompted. We will outfit the tank and wand with infrared light-emitting diodes (IR LEDs) and use an OV7670 camera with a visible-light filter to capture each item's position. This information is then sent to the Nexys 4 FPGA to be processed into commands for the tank's RC controller, which will guide the tank along the wand's route.

2 Design



The system will operate in two main modes. In "record" mode – entered by hitting a button on the Nexys 4 FPGA – the system seeks to store the wand's coordinates as a path for the tank to eventually follow. In this mode, the tank is stationary, and the Wand Coordinates Buffer is constantly updated.

The “follow” mode (entered after “record” mode by pressing the same button on the Nexys 4) uses the camera’s live feed to determine the tank’s location, and compares this to the stored wand coordinates. From these coordinates, vector data is calculated to later create appropriate commands for the tank’s remote control.

3 Implementation

3.1 OV7670 Camera

A camera is outfitted with a visible-light filter that will only allow the IR LEDs to be captured. This data will be stored in BRAM while it is processed by the Coordinates Filter. For debugging purposes, the live feed from the camera may also be directly output to an LCD display.

3.2 Signal Conversion

The raw output from the camera will need to be modified into a suitable format before it can be further manipulated or displayed. This module will handle the conversion of the camera’s data lines and provide the Coordinates Filter with a recognizable set of VGA signals.

3.3 Coordinate Finder

This module receives the VGA-compliant capture data from the conversion module. By convolving this data with a skeletonization kernel, the locations of the IR LEDs will be further isolated. This skeletonized image will be stored in BRAM as it is processed, and will be scanned using a thresholding filter to determine the exact pixel coordinates of each IR LED. Each “coordinate” will be 20 bits long: the upper ten will describe the x-position of the LED, and the lower ten bits will describe the y-position. When the coordinates have been determined, they will be forwarded to the Coordinate Separator.

To test this module, we’ll display the live camera feed and overlay the system’s guess (in the form of a sprite) for each LED location on a lab LCD. The module succeeds if the camera feed and output coordinates align.

3.4 Coordinate Separator

After receiving the coordinates of each IR LED, the Coordinate Separator will determine which LEDs correspond to the tank, and which is the wand. This will initially be completed using a mathematical algorithm to determine which sets of LEDs are arranged in the pattern of an isosceles triangle (how they'll be positioned on top of the tank). These coordinates will then be further distinguished by their locations on top of the tank (front, back left, back right), and output to the Error Signal Generator when prompted.

The coordinates of the wand LED will be stored in BRAM frame by frame.

To test the processor, we’ll again overlay the system’s LED location guess with the live camera feed, but we’ll also distinguish the output coordinate guesses by using colored sprites. The sprites will indicate the tank’s front LED, back LEDs, and the current position of the wand.

3.5 Wand Coord BRAM

The BRAM for the wand coordinates will be written to by the Coordinate Separator while recording, and read from by the Error Signal Generator after recording has finished. Its size will be sufficient to hold 15 seconds of data (15 x 30 FPS => 450 spaces holding 20 bits each).

3.6 Error Signal Generator

The purpose of this module is to translate those coordinates into useful error signals which the controller can correct for by moving the tank. This is is done using feedback. If the project is a system with a controller and a plant, then the RC tank is the plant, and its output is its position. In order to produce stable behavior from the plant, the camera tracks the current coordinates of the RC tank, and those are continuously fed into the Error Signal Generator. The "error signals" will be the distance from the front LED of the car to the next point on the path (11 bits), as well as the angle between the forward vector of the tank and the vector pointing from the tank to the next point on the path (9 bits).

Calculating the distance is as straightforward as applying the Pythagorean distance formula. To calculate the angle, we will define a new coordinate system using the midpoint of the line connecting the back LEDs of the tank as the origin. Then we will convert the Cartesian coordinates of the front LED and the next point on the path to polar coordinates and take the difference between their angles. The distance formula will require multiplication, addition, and square root operations, while the angle calculation will require inverse tangent, division, and subtraction. Both parameters will be expensive in terms of clock cycles, however time is not an issue here because we only need to meet the refresh rate of the camera, which is 30Hz. We will test this module by passing in various coordinate combinations for the tank and path using the on-board switches, and display the outputs using the hex displays on the Nexys 4. We can compare those values to our own hand calculations to confirm the module works properly.

As one of our stretch goals, we will eliminate the need for the IR LED's and filter by directly identifying the tank and wand using correlation kernel techniques. This will decrease the physical setup time for our system, and of course reduce cost.

3.7 Speed Controller

This module will take in the angle between the tank and the next point in the path (9 bits), and the distance from the front LED of the tank to that point (11 bits). Its purpose is to convert these measurements into two values between 0 and 255 (using 8 bits each) which will represent the desired linear and angular velocity for the tank. Our first pass of this module will involve implementing a simple proportional controller, which means each input will be shifted to the right and then multiplied by a constant. As with the last module, timing is not an issue, so we don't need to worry about the propagation delays of multiplies.

The testing for this module will also be similar to the last. We will feed in various values for the measured distance and angle, display the outputs on the hex displays, and check to make sure they match what we expect. The constants used for the multiplication will be parameterized and, once we are able to test all of our modules working together, we will use the on-board switches to change their values and find those that give the best performance.

A proportional controller is our minimum goal for this module. Once our full project is operational, we will improve the tank's performance by implementing a proportional-integral controller, and then a proportional-integral-derivative controller.

3.8 Car Control Converter

In order to produce the desired behavior from the tank, we need to send our control signals in the appropriate format. This module takes in the values between 0-255 (8 bits) for the linear and angular speeds and converts them into that format. However, the specific format needed is dependent on how we interface with the tank. Our first option is to hijack the transmitter that comes with the tank, and use the I/O pins of the FPGA to simulate button presses. Depending on the design of the transmitter and the electronics on the tank, this method might not provide enough accuracy for us to stably control the tank. Many RC cars use chips that wait for a certain number of square pulses at a specific frequency to be received, and then switch between full power/no power. This control system makes accurate acceleration control difficult.

Alternatively, we will directly mount an XBEE onto the tank and connect its GPIO pins, through an H-bridge, to the tank's motors. The on-board tank battery will be used to power the XBEE. This XBEE will act as a receiver, and another XBEE, connected to the FPGA, will send out the desired PWM for the motors. In this scenario, this module only needs to convert the inputs to the corresponding duty cycles for the PWM, and interface with the XBEE. We will test this module by setting the linear velocity equal to the scaled value of the Nexys 4's "UP" push button and similarly setting the angular velocity equal to the scaled values of the left and right push buttons. Either way, the physical reaction time of the tank is relatively slow, so we will zero order hold the control signals between each rising edge of the 30 Hz camera clock.

3.9 Shape Recognition module

As one of our stretch goals, we want to implement functionality that will allow the user to command the tank to execute a pre-defined command, such as spinning in place. The user will choose which command to execute by drawing the corresponding shape for that command with the wand when the system is in a specific standby mode controlled by one of the buttons on the FPGA. The purpose of this module is to read in the array of coordinates of the wand that were stored while the user was drawing the shape, then determine which command will be executed. We will have two of these built-in commands. One will be executed by drawing a circle, the other by drawing a line perpendicular to the direction the tank is facing, in front of the tank. This module will output either a one or a zero, which will let the top-level module know which command to execute.

4 Timeline

- Week of Oct. 30
  - Car Control Converter, Camera Display
    - Here we create a top-level module to write frames from the camera to BRAM and create a module to read those frames and display them on a screen. We will design a module to interface with the transmitter that comes with the RC tank and allow the user to drive the tank using buttons on the FPGA. The module that displays the camera live feed will be the responsibility of Andre. Druck will be responsible for the module which interfaces with the transmitter and controls tank operation. Both members will be responsible for the physical interface of the camera and tank with FPGA.
- Week of Nov. 6
  - Speed controller/Error Signal Generator, Coordinate Finder
    - In this stage we will finish the module which reads video frames from BRAM and processes them to pinpoint the locations of the IR LEDs and we will create the modules that take in those coordinates and calculate the appropriate response of the tank. Druck will be responsible for the speed controller and error signal generator, while Andre will be responsible for the coordinate finder.
- Week of Nov. 13
  - Integrate ESG SC and CCC, Coordinate Separator
    - At this point we will combine the error signal generator, speed controller, and tank controls converter into one top-level module so that we can test our system's ability to move the tank in the direction of a point in our coordinate system. We will also finish the module which differentiates and labels each LED. Creating the top level module for the tank controller will be Druck's responsibility while creating the coordinate separator module will be Andre's responsibility.
- Week of Nov. 20
  - Top Level Module, Testing/Debugging, Shape Recognition Module
    - At this point we will be finished implementing our bare-minimum modules and will create the top level module for our project. We will also begin work on additional functionalities, by completing the module which recognizes shapes drawn by the user. Andre will create the top-level module for the project while Druck creates the shape recognition module. Both members will setup and test the functionality of the top-level module.

- Week of Nov. 27
    - Finish Testing, Display Path, Implement Tricks Mode, PID Speed Controller
        - Here we will finish debugging the top-level module and sub modules needed for minimum functionality. We will also show the user how the FPGA is tracking the path and tank by displaying sprites on the screen. The shape recognition module will tell the error signal generator to pull from a different BRAM where the special paths are located. Andre will be responsible for displaying the path drawn by the user and the location of the cart as sprites. Druck will wire the top-level module to implement the tricks mode, and upgrade the proportional speed controller to a PID controller.
- Week of Dec. 4
    - Testing/Debugging
        - We will use any remaining time to finish testing the system as a whole and fix any bugs and/or optimize performance. This will be done by both team members.

5 Responsibilities

Interfacing with the tank and camera are essential to the completion and understanding of the project and will be the responsibility of both members. The image processing, detection, and displaying modules will be Andre's responsibility. The tank control, and shape recognition modules will be Druck's responsibility .

6 Resources

To implement this design, we'll need to procure:

- ~10 Infrared LEDs
- RC Tank
- OV7670 Camera
- Nexys 4 FPGA
- Battery pack
- Floppy Disk (contains the film that will be used as the IR filter for the Camera)

7 Conclusion

The LED Pathfinder is an exciting project for our team. We chose to complete this project because of our interest in learning to interface with different types of hardware. Hacking an RC tank and video camera provides ample opportunity to work around real-world non-ideal aspects in technology. We predict a fun, but challenging experience.