# Duckhunter

6.111 Final Project Proposal

Taylor Barton and Andrew Lisy

November 4, 2005

**Abstract**

*Duck Hunt* is a classic first-person shooter game for the Nintendo Entertainment System, in which the player uses a special light gun to "shoot" images of either one or two ducks or clay pigeons, depending on the game mode. Our *Duckhunter* project will focus on the third mode of this game: clay shooting. In this mode, two clay pigeons are launched from the bottom of the screen into the distance, and the player is given three shots to hit them both before they are too far away.

In order to beat this game, we will build a servomechanism that will automatically track and shoot down the clay pigeons. The labkit's FPGA will be programmed to "look" at the video output of the Nintendo System by receiving its video outputs, find the moving targets, and calculate the angle at which to aim the light gun. It will output the appropriate signals to two servomechanisms that will control up-down and left-right aiming, and fire the light gun. Once this basic function has been implemented, we will demonstrate the speed of the FPGA by playing our *Duckhunter* against either an overclocked Nintendo System or an emulater running the game at a high speed.

# 1 Description

Our project, Duckhunter, is an FSM-controlled robot that will play the game *Duck Hunt*, made popular on the Nintendo Entertainment System in the early 1990s. The game itself has three modes: single duck, double duck, and clay pigeon (skeet shooting). Our project will focus on the third mode and implement a system to track, predict, aim and fire the gun assembly.

The project itself is composed of three major components: the video tracking system, the servo control system and the calibration system. These modules will interface with each other via a central FSM which will process the inputs and outputs based on the current state.

The first system, video tracking, will be responsible for determining the coordinates $(x_1, y_1)$ and $(x_2, y_2)$ of the two moving clay pigeons. To accomplish the tracking, the system will look at the initial velocity vector $\langle \overrightarrow{x_0}, \overrightarrow{y_0} \rangle$ and initial intercept, $x_0$, from which the disc is shot. It will then apply formulas determined by testing the Nintendo physics engine to calculate a position equation based on the intercept, $x$, initial upward velocity $\overrightarrow{y}$ and initial horizontal velocity $\overrightarrow{x}$. From these formulas, it will determine the predictors for each of the discs.

The servo control system will handle the translation of an $(x, y)$ coordinate into the appropriate aiming of the gun. Since a module is provided to send the actual position to the servos, this module will be concerned primarily with translating a coordinate into the appropriate positions for the servos controlling the horizontal $\theta_h$ and vertical $\theta_v$ angles of the aiming assembly. It will be implemented using constant values obtained for the overall motor aiming system plus a calibration component $\phi$ determined at the beginning of each runtime by the calibration system.

Finally, the calibration system enables the gun assembly to be moved, stored, then reattached to the system without lengthy manual recalibration. Since the assembly will not be in the same position from run to run, it is necessarily to determine and compensate for differences in the positioning over different trials. Upon repositioning the gun, the calibration system will run to determine the calibration components for the current trial. The routine will be implemented using the aiming and timing subsystems described in the next section.

# 2 Implementation and Testing

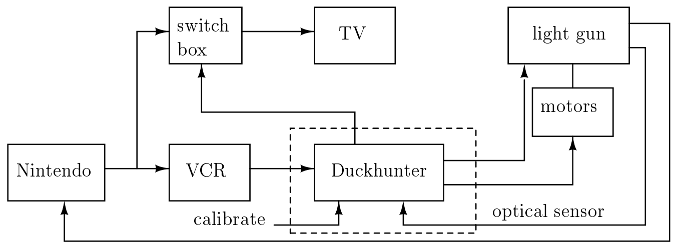
The block diagrams for the implementation are given in Figure 1. Figure 1(a) shows how the overall system will be put together. The input to this system is generated by the Nintendo System. Ordinarily, the Nintendo's video output is fed into the TV, which displays the Duck Hunt game. Additionally, the video is used as the input to a VCR, which converts it to a RCA video signal that the labkit can use. Duckhunter's other input is a signal from the light gun which is used for calibration and aiming. Its outputs are aiming and shooting signals to the light gun, as well as a video output signal that can be used instead of the Nintendo's video output for calibration and testing.

A detailed diagram of the Duckhunter system is shown in Figure 1(b). It is composed of a collection of modules, which are explained individually below. Methods to test each block are also discussed below.

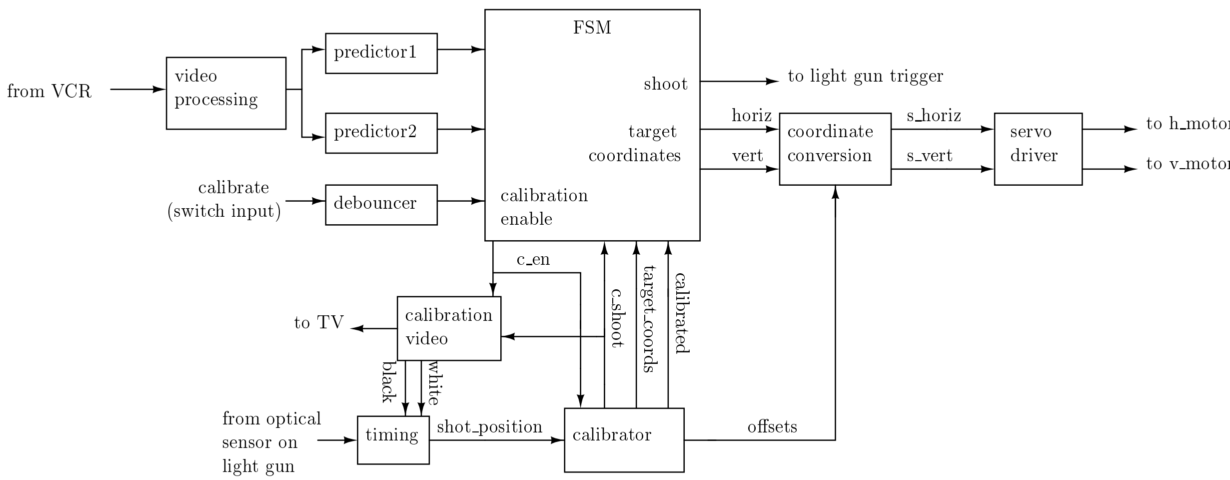
## 2.1 Video Processing

The video processing block takes the RCA input from the VCR as its input. It subtracts the background image from the video input, leaving only the white targets on a black background. This subtraction is straightforward because the background for Duck Hunt's skeet mode is constant, so this image only needs to be found and stored once.

This module is simple to test, as its video output can be fed directly to a monitor, and the images compared with the TV images from the Nintendo System.




(a) System Block Diagram




(b) Duckhunter Block Diagram

Figure 1: Block Diagrams for Duckhunter

## 2.2 Predictors

The two predictors take in the video output of the video processing block and use it to locate and track one of the two targets on screen. Using physics (as defined by the Nintendo System), they solve the equation of motion of the targets. The output of each predictor is the coordinate where one of the targets will be after a certain constant aiming delay. This delay will be determined during testing.

The predictors can be tested by generating a known image with a known trajectory and checking that the output coordinates fit the equation of motion. An example input would be from a module similar to the Pong game implemented in Lab 4, with a white puck on a black background that moves parabolically rather than in a straight line.

## 2.3 Finite State Machine (FSM)

The finite state machine's basic function is to take the target coordinate inputs, figure out which coordinate to shoot at, and activate the light gun trigger. It also interfaces with the calibration system through its calibration enable output, which is activated by the switch input on the labkit. The finite state machine additionally takes in position and trigger signals from the calibrator module. If the finite state machine is in the calibration state, then these inputs are routed directly to the target coordinate and shoot outputs of the finite state machine. That is, when in the calibration state the finite state machine ignores the video input from the Nintendo system, reacting instead to the video signals provided by the calibration video module.

### 2.4 Coordinate Conversion

The coordinate conversion module uses simple math to convert from the $(x, y)$ coordinates from the finite state machine into angle coordinates for the motors. The exact details of this computation will be set by parameters that depend on the distance of the gun from the TV. Additionally, the coordinate conversion module takes horizontal and vertical offset signals, which are outputs of the calibrator module. These offsets are used to fine-tune the aiming of the light gun.

The coordinate conversion module is straightforward to test; simply give it known inputs and check that the outputs correspond as expected.

### 2.5 Servo Driver

The driver for the servomechanism takes in two 8-bit position signals and drives the motors with pulses so they reach the right position. Its output goes directly to the two RC motors. The driver can be tested simply by giving it a position input and watching the motors move.

### 2.6 Calibration Modules

There are two common methods used by Ninendo and similar Systems to determine whether the light gun shot hits the target.[1] The computer first blanks the screen when the trigger is pulled. For the first method, it then colors the target object location white, keeping the rest of the screen black. If the photosensor senses the white area, then the shot was on target. In the second approach, the entire screen is made white after the blanking. Because of the time it takes to change each pixel, the position of the shot can be determined by the delay between the start of coloring the screen white and the time at which the photosensor senses white.

The exact use for the following modules depends on which of these two methods the Duck Hunt game uses. If it uses the latter, the calibrator will be useful not only for a calibration cycle prior to game play, but also for continuous re-calibration after each shot.

#### 2.6.1 Calibrator

The calibrater module serves two functions. First, it is active during the initial calibration mode, which runs prior to playing the game to make sure the aiming is aligned. If the calibration enable input is high, the calibrator aims the gun at a particular point using the target_coords output, and fires it using c_shoot. Through the timing module, it then gets the input shot_position corresponding to where the shot hit. If the intended and actual coordinates are different, this module takes the difference for horizontal and vertical coordinates and gives this output to the coordinate conversion. This process is repeated until the offset is set such that the intended coordinates and actual coordinates are the same. Once the system has been calibrated, the calibrator module output goes high.

The second function of this calibrator module is to serve as a continuous calibrator throughout game play. This function will only be possible if the Duck Hunt game uses the all-white targeting method described above. Once the Nintendo System and game have been acquired, it will be clear whether this function is a possibility.

Since this module is by definition a test module, it is very easy to test. The shot_position input can be generated independently of the timing module working.

[1] http://entertainment.howstuffworks.com/question273.htm

### 2.6.2 Calibration Video

The calibration video module produces a sample video output to test and calibrate the aiming system. It inputs are c_en, the enable input; and c_shot, an input indicating that the calibrator module has pulled the light gun's trigger. When both of these inputs are high, this module first blanks the screen and brings its black output high, and then colors the screen white while simultaneously bringing its white output high. these signals are used by the timing module to deduce the position of the shot.

### 2.6.3 Timing Block

Upon receiving a high black input, the timing module stores the output of the photosensor on the light gun, since this value corresponds to the value of sensing black. When its white input is high, the module counts the number of clock cycles between white going high and the photosensor detecting white. The timing block determines when the sensor sees white by comparing its output with the stored value for a black screen. By comparing this time with the horizontal and vertical retrace signals (not shown on the block diagram to reduce clutter) it can determine the position shot. This shot position is the output of the module.

The timing block module can be tested by pointing the gun at known positions – the four corners and the center, for example – and checking that its output is the correct coordinates.

# 3 Work Division

Taylor Barton will design and implement the calibration system and the servo system. From the block diagram, she will build and test the timing, calibrator, calibrator video, coordinate conversion and servo driver subsystems.

Andrew Lisy will design and implement the video tracking system and the FSM. From the block diagram, he will build and test the video processing, predictor 1, predictor 2 and FSM subsystems.