CENG3430 Final Project Report

Ian Ha Jin Quan (1155138078)

Contents:

Introduction

I have taken an interest in the use of gun accessories/controllers in games, Nintendo in particular, released a light gun called NES Zapper that was launched alongside the NES in 1985. This system does not track the position of the gun by gyroscope or accelerometer, but rather, they implemented a simple passive system which only make use of a light sensor to determine whether the gun is pointing at the target every time the player pull the trigger.

This system is not only simple to implement, as it does not have to deal with the calibration and drifting issue of the gyroscope based system(particularly cheaper ones), but also able achieve very high accuracy with relatively low cost (cheap light sensor). This project aims to implement a simplified version of the NES game “Duck Hunt” that makes use of this light gun system.

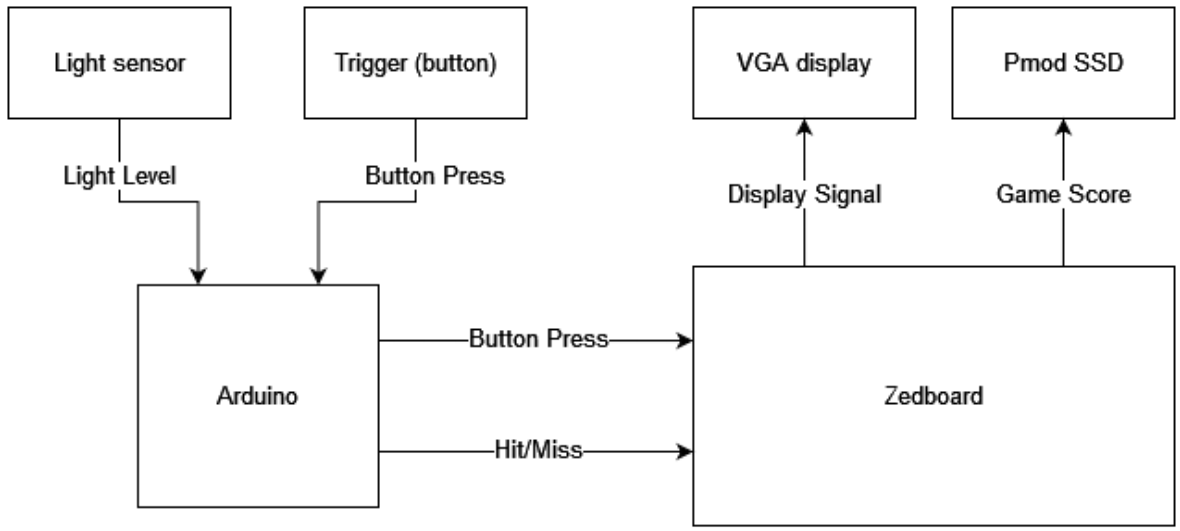
Design

Components and IO:

This project will be implemented using VHDL and Arduino.

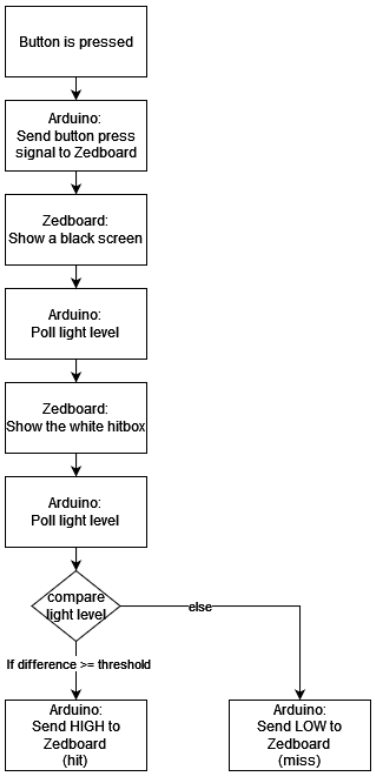
The Arduino will take the inputs from the light sensor and buttons, and determine whether the target is hit or missed based on the light level. It will output the button press and the hit/miss signal to the Zedboard.

The Zedboard will receive the button press input and hit/miss signal from the Arduino, outputs the game screen via the VGA port, and the game score to the Pmod SSD.



Mechanics of Light Gun:

To determine whether the player hits the target on screen, when the player press the button, the screen will first show a black screen for a short period of time. This will allow the light sensor to poll the black level. Then, the screen will show the white hitbox on the position of the bird. The sensor will poll the light level again. If the gun is pointed at the bird, there will be a difference in two light levels, and the system knows that the target is hit. Otherwise, if the gun is not pointed at the bird when the button is pressed, the two light levels will be very close since the light sensor will be polling the black part of the screen two times, and the system knows that the target is missed



Game Flow:

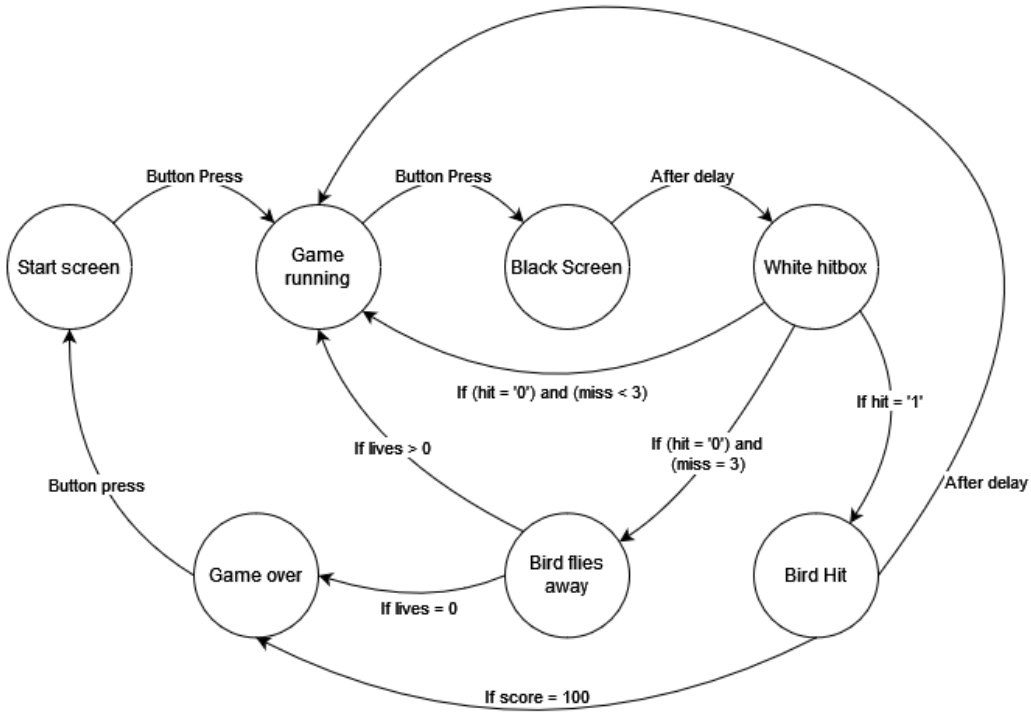
The game starts on the start screen. When the button is pressed, the game will start running and a bird will spawn from the bottom of the screen at a random x-position. When the button is pressed (shooting the gun), the screen goes black momentarily and flashed a white box afterwards. If target is hit, the bird will turn red and the score will increase. After that, another bird will spawn again from the bottom. If the target is missed, the bird will continue flying. After it has been missed 3 times, the bird will fly away, and another bird will spawn.

Initially when the game starts, the player will have 9 lives. Each miss will cost the player one life, and the it will be game over once the player runs out of lives.

Initially, the bird will move slowly, but for each 10 birds spawn, the game will enter next “stage”, the bird will speed up, and the player will gain one additional life after each “stage”.

When the player reaches 99 points, the game will be over.

FSM of the game:



Implementation

The states are represented by integer in the VHDL code, and the game runs on the 100hz clock.

State 0: Game Running

```
begin
    if rising_edge(clk100Hz) then
        if (lives = 0) then
            state <= 101;
        end if;
        if (state = 0) then          --game running
            X_STEP <= 2 + (bird_count/10)*2;
            Y_STEP <= 2 + (bird_count/10)*2;
            if (edge_detect = "01") then
                state <= 1;
            else
                if(x + WIDTH >= H_END) then
                    dx <= -X_STEP;
                end if;
                if(x <= H_START) then
                    dx <= X_STEP;
                end if;
                if(y + HEIGHT >= V_END) then
                    dy <= -Y_STEP;
                end if;
                if(y <= V_START) then
                    dy <= Y_STEP;
                end if;
                    x <= x + dx;
                    y <= y + dy;
                end if;
```

Calculate the speed of the bird

Detect rising edge of button

Position of the bird

State 1: Black Screen

```
elsif (state = 1) then       --black calibration screen
    if (count <= 4) then
        count := count + 1;
    else
        count := 0;
        state <= 2;
    end if;
```

Clock counter

Transition to another state after certain clock count

State 2: White hitbox

```
elsif (state = 2) then       --white hitbox
    if (count <= 2) then
        count := count + 1;
    else
        count := 0;
        if (hit = '1') then
            score <= score + 1;
            miss <= 0;
            if (score >= 99) then
                score <= 0;
            end if;
            state <= 3;
        else
            miss <= miss + 1;
            lives <= lives - 1;
            if (miss >= 2) then
               miss <= 0;
               state <= 4;
            else
                state <= 0;
            end if;
        end if;
    end if;
```

When hit, increment score, reset miss count

Reset score (ssd can only display 2 digits)

When miss, increment miss, decrement lives

If miss 3 times, bird flies away

State 3: Bird hit

```
elsif (state = 3) then       --bird gets hit
    if (count <= 100) then
        count := count + 1;
    else
        count := 0;
        if (score >= 99) then
            state <= 101;
        else
            bird_count := bird_count + 1;
            if ( (bird_count > 1) and (bird_count mod 10 = 0 )) then
                lives <= lives + 1;
            end if;
            x <= (rand mod (H_ACTIVE)) + H_START;
            y <= V_END;
            state <= 0;
        end if;
    end if;
```

If score reaches 99, game over

Increment lives for each 10 birds spawned

Spawn bird at random x-position

State 4: Bird flies away

```
elsif (state = 4) then      --bird flies away
    y <= y - 4;
    if (y + HEIGHT <= V_START) then
        count := count + 1;
        if (count = 100) then
            count := 0;
            bird_count := bird_count + 1;
            if ( (bird_count > 1) and (bird_count mod 10 = 0 )) then
                lives <= lives + 1;
            end if;
            x <= (rand mod (H_ACTIVE)) + H_START;
            y <= V_END;
            state <= 0;
        end if;
    end if;
```

Bird flies away vertically upwards

Wait until the bird flies away from the screen

Increment bird count

State 100: Start Screen

```
elsif (state = 100) then      --start screen
    if (edge_detect = "01") then
        lives <= 9;
        miss <= 0;
        dx <= 2;
        dy <= 2;
        bird_count := 0;
        score <= 0;
        x <= (rand mod (H_ACTIVE)) + H_START;
        y <= V_END;
        state <= 0;
    end if;
```

Detect rising edge of button

Reset game variables

State 101: Game over

```
elsif (state = 101) then      --game over screen
    if (edge_detect = "01") then
        lives <= 9;
        miss <= 0;
        dx <= 2;
        dy <= 2;
        score <= 0;
        x <= (rand mod (H_ACTIVE)) + H_START;
        y <= V_END;
        state <= 100;
    end if;
```

Detect rising edge of button

Reset game variables

Random number generator:

```
--rng
process (clk100Hz)
variable clock_count : integer := 0;
begin
    if rising_edge(clk100Hz) then
        clock_count := clock_count + 1;
        if (clock_count >= 2000000) then
            clock_count := 0;
        end if;
        rand <= clock_count * 13 + 123;
    end if;
end process;
```

Display graphics:

```
process (hcount, vcount, x, y)
begin
if (state < 100) then
    if ((hcount >= H_START and hcount < H_END) and (vcount >= V_START and vcount < V_TOTAL)) then
        if (x <= hcount and hcount < x + WIDTH and y <= vcount and vcount < y + HEIGHT) then
            if (state = 0 or state = 4) then
                if (dx > 0) then
                color <= fig((vcount-y)/8, (hcount-x)/8);
                else
                color <= fig((vcount-y)/8, (x + WIDTH - hcount)/8);
                end if;
            elsif (state = 1) then
                color <= C_Black;
            elsif (state = 2) then
                color <= C_White;
            elsif (state = 3) then
                if (dx > 0) then
                    if (fig((vcount-y)/8, (hcount-x)/8) /= C_BLACK) then
                        color <= C_Red;
                    else
                        color <= C_Black;
                    end if;
                else
                    if (fig((vcount-y)/8, (x + WIDTH - hcount)/8) /= C_BLACK) then
                        color <= C_Red;
                    else
                        color <= C_Black;
                    end if;
                end if;
            end if;
        else
            color <= C_Black;
        end if;
    else
        color <= C_Black;
    end if;
 else
    if ((hcount >= 747 and hcount < 851) and (vcount >= 252 and vcount < 348)) then
        if (state = 100) then
            color <= fig((vcount-252)/8, (hcount-747)/8);
        elsif (state = 101) then
            if (fig((vcount-252)/8, (hcount-747)/8) /= C_BLACK) then
                color <= C_White;
            else
                color <= C_Black;
            end if;
        end if;
    else
    color <= C_Black;
    end if;
 end if;
end process;
```

Show bird figure

Flip bird figure when going negative direction

Display the bird in red

Display static bird figure

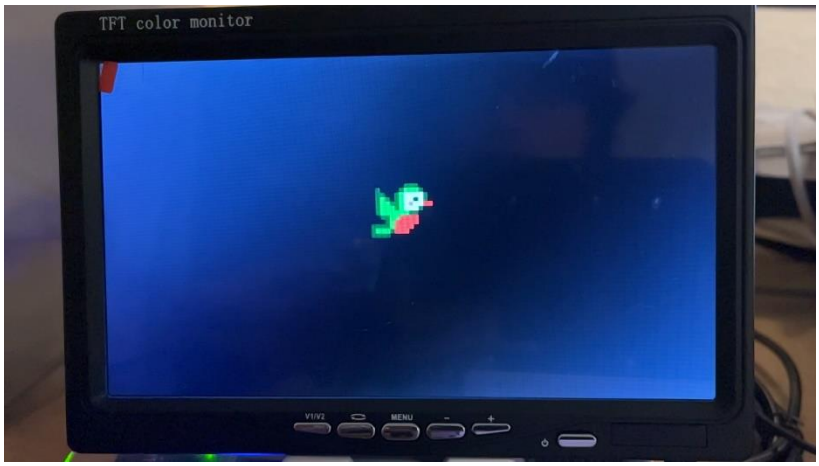
Display static bird figure in white

Arduino Code:

```
void loop() {
  int buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH){
    digitalWrite(buttonOut, HIGH); //send signal to Zedboard

    delay(50); //input lag

    //calibrate dark level
    int cal = analogRead(A0);
    Serial.print("cal: ");
    Serial.print(cal);

    delay(interval);

    //sample light level
    int hitscan = analogRead(A0);
    Serial.print(" hitscan: ");
    Serial.println(hitscan);

    Serial.println(cal - hitscan);

    //if hit
    if((cal-hitscan) >= threshold){
      digitalWrite(hit, HIGH);
      delay(150);
      digitalWrite(hit, LOW);
    }else { //if miss
      digitalWrite(hit, LOW);
      delay(150);
    }
    delay(10);
  } else {
    digitalWrite(buttonOut, LOW);
  }
}
```

Results

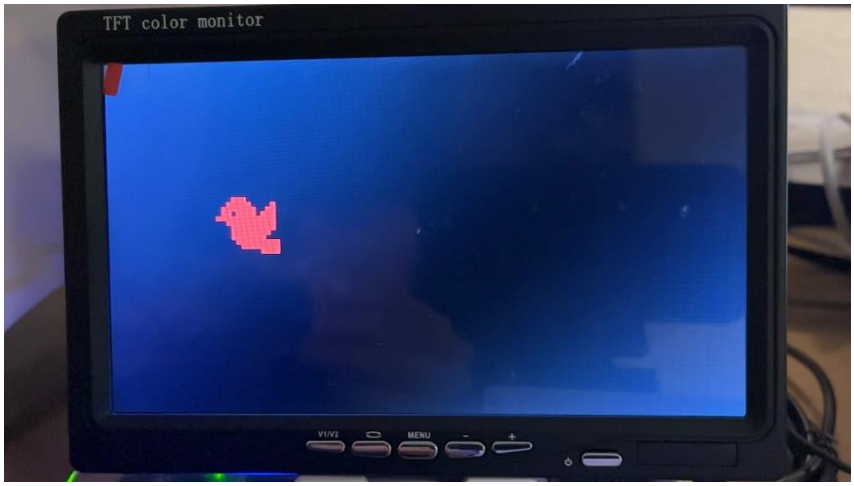
Start screen:



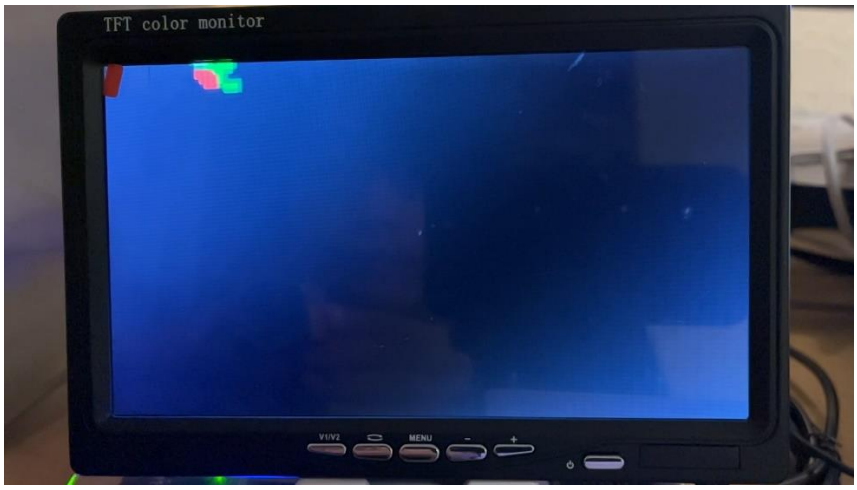
Black screen:

White hitbox:

Bird hit:



Bird flies away:



Game over screen:

Difficulties

The most difficult part of this project is to get the timing right. Timing is very critical in this system because the light levels will only be polled twice (instead of continuously polling), and if the light level are polled at the wrong time, this will lead to unreliable and inconsistent hit detection.

There are several delays that should be considered in this system:

1. Delay between the button press and the screen starts to go black (input lag)
2. Delay between the screen at the darkest level and the white hitbox at the brightest level.
3. Delay for the pixel to change color, i.e. the time from the pixel starting to change color to the time where the pixel finish changing color (response time)

To find out the exact timing, I filmed a high-speed video of the screen at 240fps and count the number of frames to calculate the input lag and the time interval between to polls.

References

Button Edge Detection: https://stackoverflow.com/a/33074498

Mechanics behind the NES Zapper: https://youtu.be/Nu-Hoj4EIjU, https://youtu.be/cu83tZIAzlA