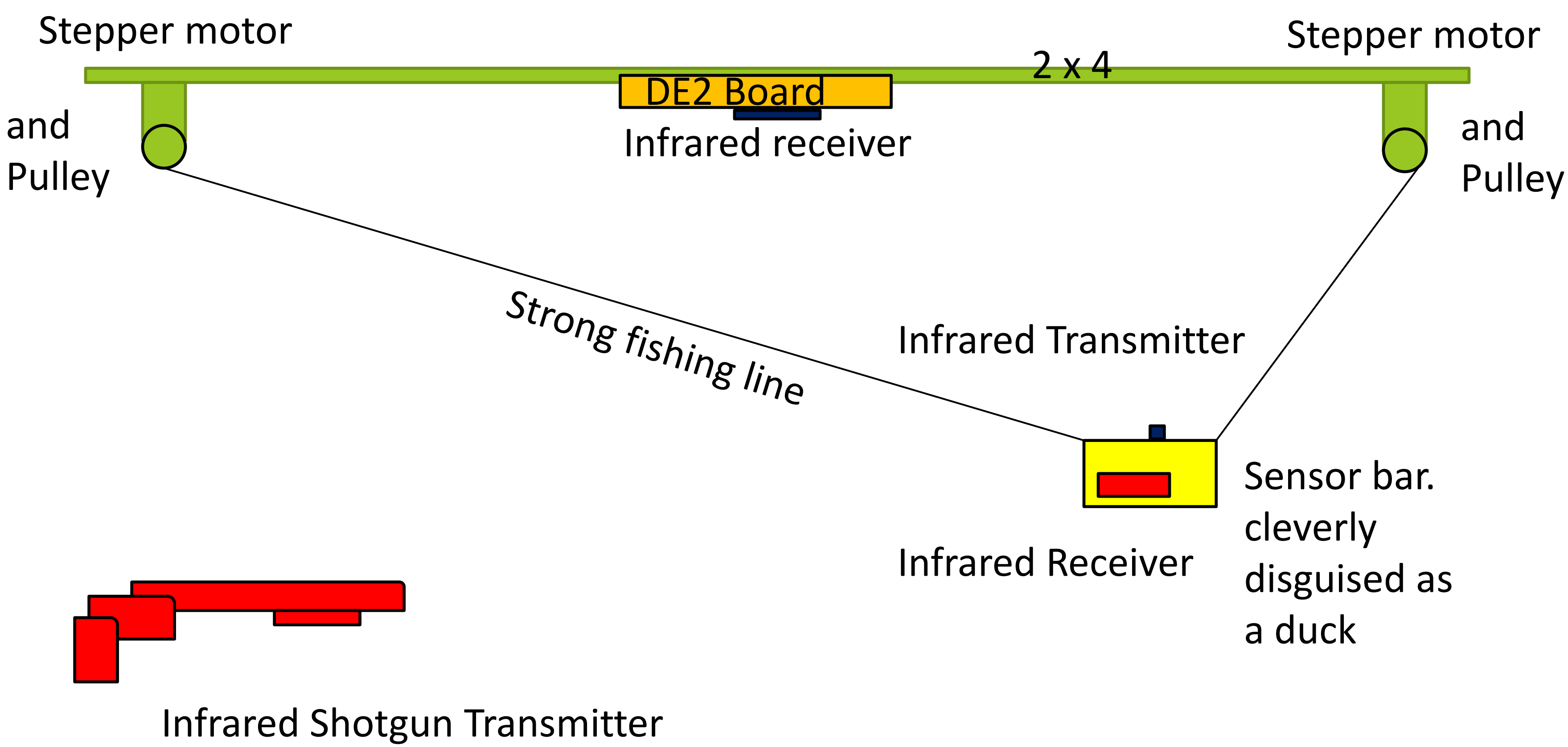
G1
Duck Hunt Game

Jesse Larson (jrlarson@ualberta.ca)
Jing Lu (jlu9@ualberta.ca)
Qingqing Liu (qliu6@ualberta.ca)

Design:
General Structure of Game Hardware



Design: General Behavior& Functionality

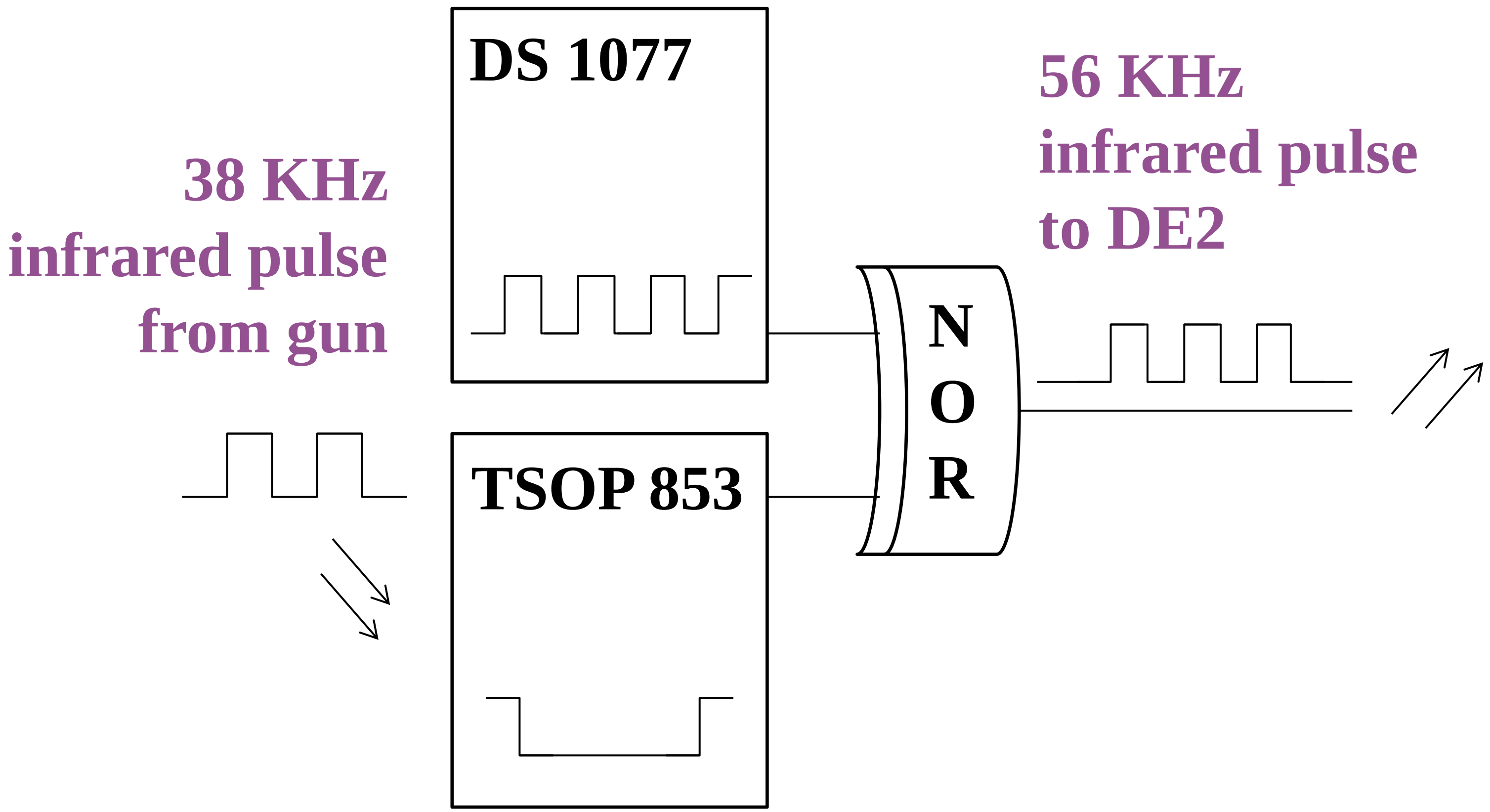
The game can distinguish between two different IR pulses. This is so that there can be two different players or two teams participating in the game.

Two modes for the game:

mode 1 – The duck must be hit 3 times by shotguns for the game to end. The team with 2 or more hits wins the game.

mode 2 – The duck is invincible for 1 minute! Shoot it as many times as you can to get a high score. Output displayed to the LCD screen.

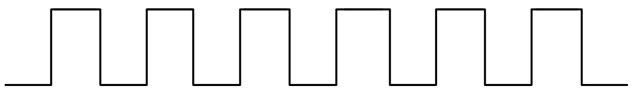
Design:
How the sensor bar works



Design:
Summary of the infrared communication challenge

Primary requirement:

The Vishay 56 KHz receiver at the DE2 board must receive a minimum of six 56KHz pulses for minimum reliable signal transfer.



However:

-The 38 KHz receiver holds its output signal low for a variable amount of time, this time dictates the number of 56 KHz pulses sent.

-The 56KHz receiver also holds its signal low for a variable amount of time as well, so the timing delay is compounded.

Design:
The solution to the IR challenge

Solution:

The solution to the delay requirements was to create a custom infrared communication protocol that achieves reliable data transfer by accommodating the delay

Brief overview of the protocol:

-Custom infrared hardware bit decoder at the DE2 board

-0 bit translates to nine 38 KHz pulses from a transmitter

-1 bit translates to twenty-seven 38 KHz pulses from a transmitter

-These are the minimum number of pulses to achieve truly reliable and distinguishable bit at the DE2 board.

Design:
The solution to the IR challenge

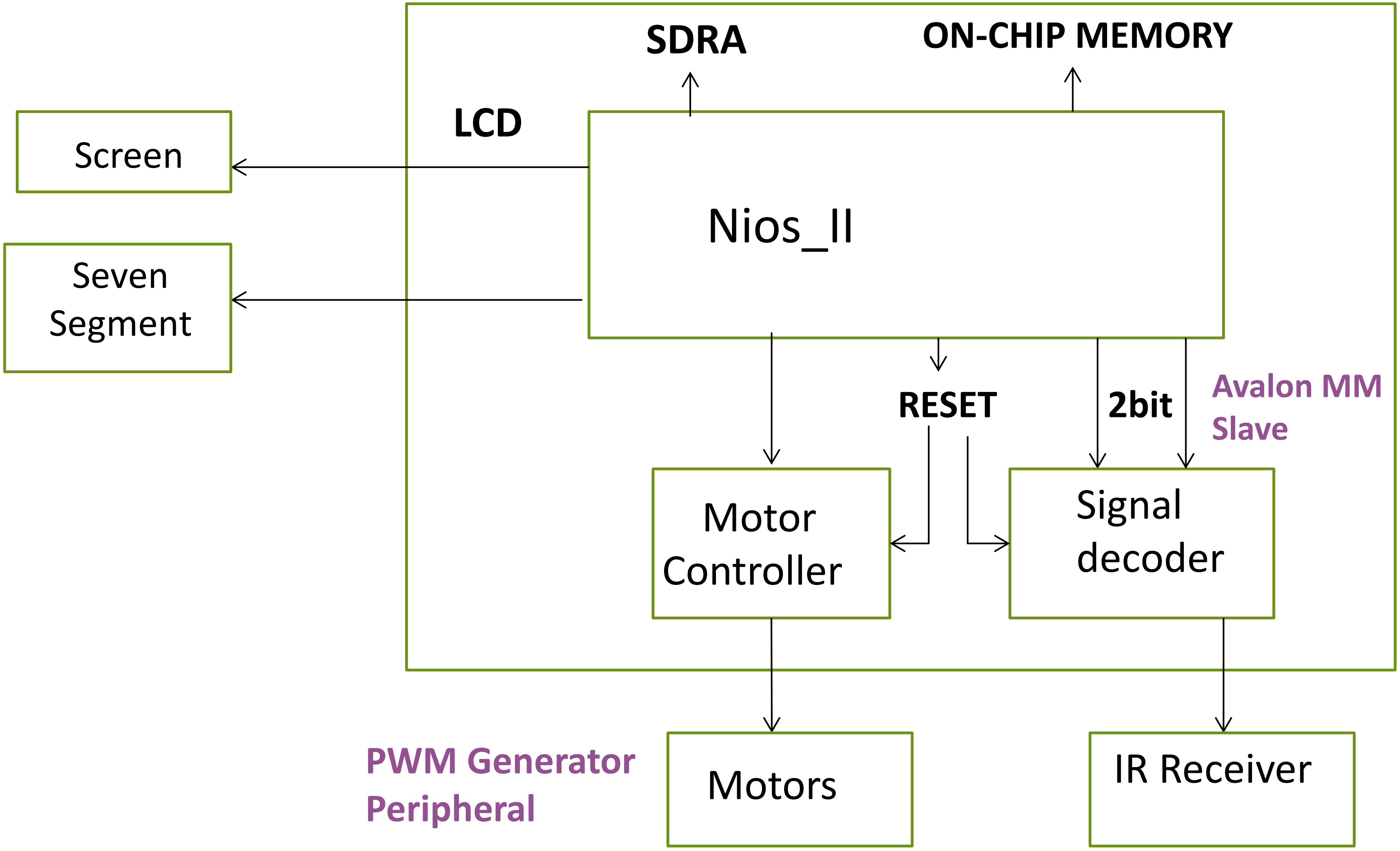
Additional challenge:

Due to poor circuit building and the sensor bar constantly being moved around on strings, the connections to ground on the sensor bar are inconsistent thus making the signal relay unreliable.

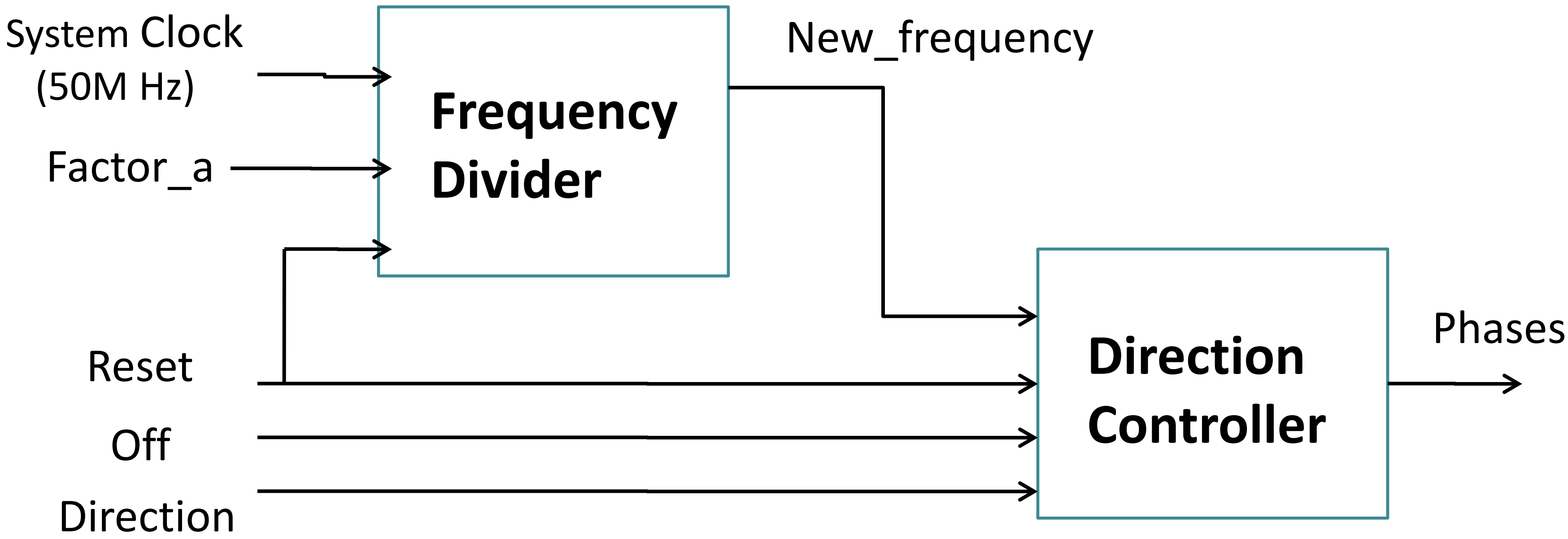
Occasionally a 1 bit fails for part of its transmission, to eliminate a some of these errors, some custom sampling hardware was added for our demo today, however it can not filter all of the errors.

Should be noted that when the circuit is properly grounded, this additional sampling hardware is never needed.

Design: DE2 Diagram



Design:
Motor Controller



Design: Frequency Divider

```
frequency_divider:process (reset,clk_in)
    begin
     if (reset='1') then
        temp<='0';
           counter<=(others=>'0');
     elsif rising_edge(clk_in) then
          if (counter>factor_a) then
            counter<=(others=>'0');
          elsif (counter=factor_a)then
            temp<=NOT (temp);
            counter<=(others=>'0');
          elsif (counter<factor_a)then
            counter<=counter+1;
        end if;
     end if;
  end process;
```

Factor_a:

Question: What is this?

Answer: A factor used for calculating the new frequency to drive motor!

Question: How do you get *factor_a*?

Answer: System frequency/(target frequency*2)-1

Example: What is the *factor_a* if I want 2000 Hz to drive my motor?

50,000,000Hz/(2000Hz*2)-1=12499

Design:
Stepper Motor

Motor in this project we used:

P1-19-4203

2 phase bipolar stepper motor

12VDC, 480mA

Coll: 25 Ohm

3.6 degrees/step

Shaft: 0.19"D x 0.43"L

Mounting Hole Spacing: 1.73"

Mounting Hole Diameter: 0.11"

Motor: 1.66"D x 1.38"H

Detent Torque: 80 g-cm

Holding Torque: 600 g-cm

Weight: 0.5 lbs.

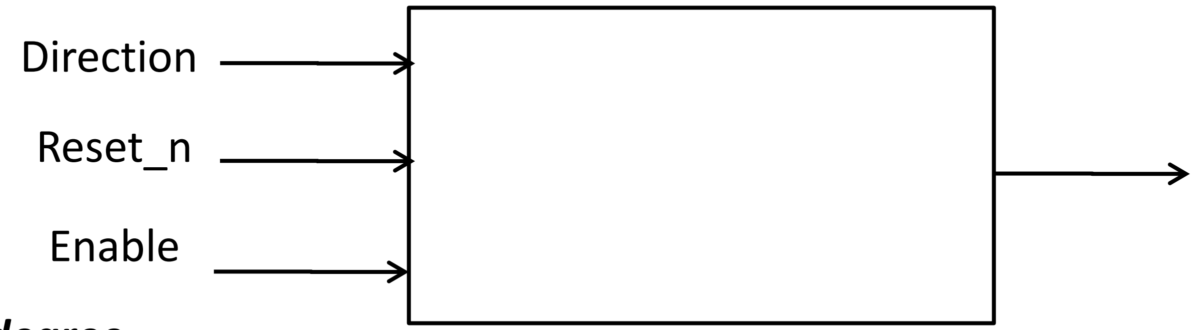
Example:

Frequency = 100Hz

Motor rotates 360 degrees in 1 second!

In our project, motor mostly run under 1Hz to 1000Hz

Design: Direction_controller



1 clock cycle=1 step=3.6 degree

Clockwise: (direction=1)
State order in 1 clock cycle:
A-----------phase = “1000”
AB---------phase = “1100”
B-----------phase = “0100”
BC---------phase = “0110”
C-----------phase = “0010”
CD---------phase = “0011”
D-----------phase = “0001”
DA---------phase = “1001”

Counterclockwise: (direction=0)
State order in 1 clock cycle:
A-----------phase = “1000”
DA---------phase = “1001”
D-----------phase = “0001”
CD---------phase = “0011”
C-----------phase = “0010”
BC---------phase = “0110”
B-----------phase = “0100”
AB---------phase = “1100”

Design:
Linear Increasing
&Gaussian random number generator

Linear Increasing:

Purpose: To avoid motor acceleration over large

which results in potentially lose step

Solution: Implement linear increasing to control the acceleration

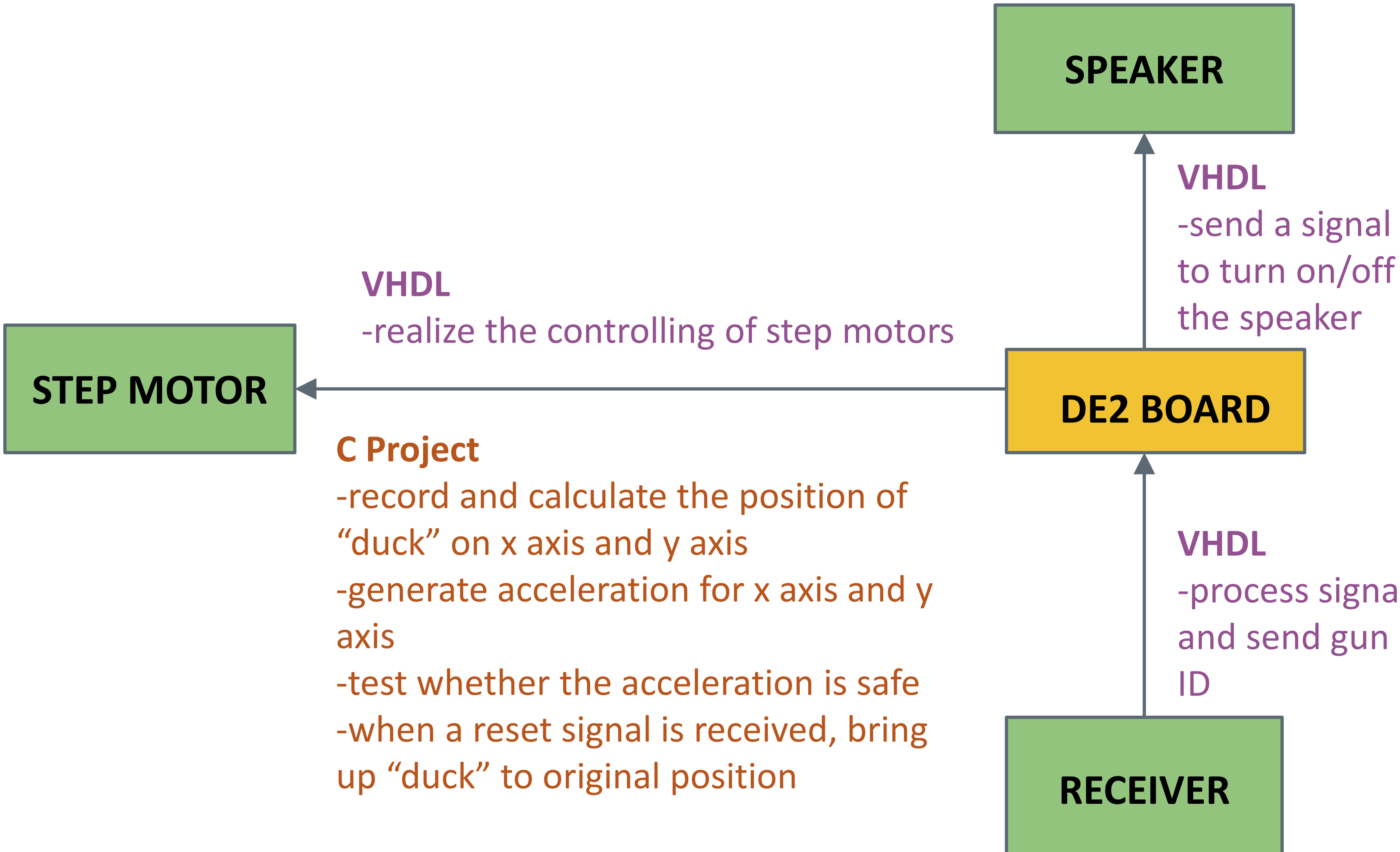
Gaussian random number generator:

Purpose: To ensure random number concentrates in the range between

1Hz and 1000Hz but still reserve the randomness that the frequency

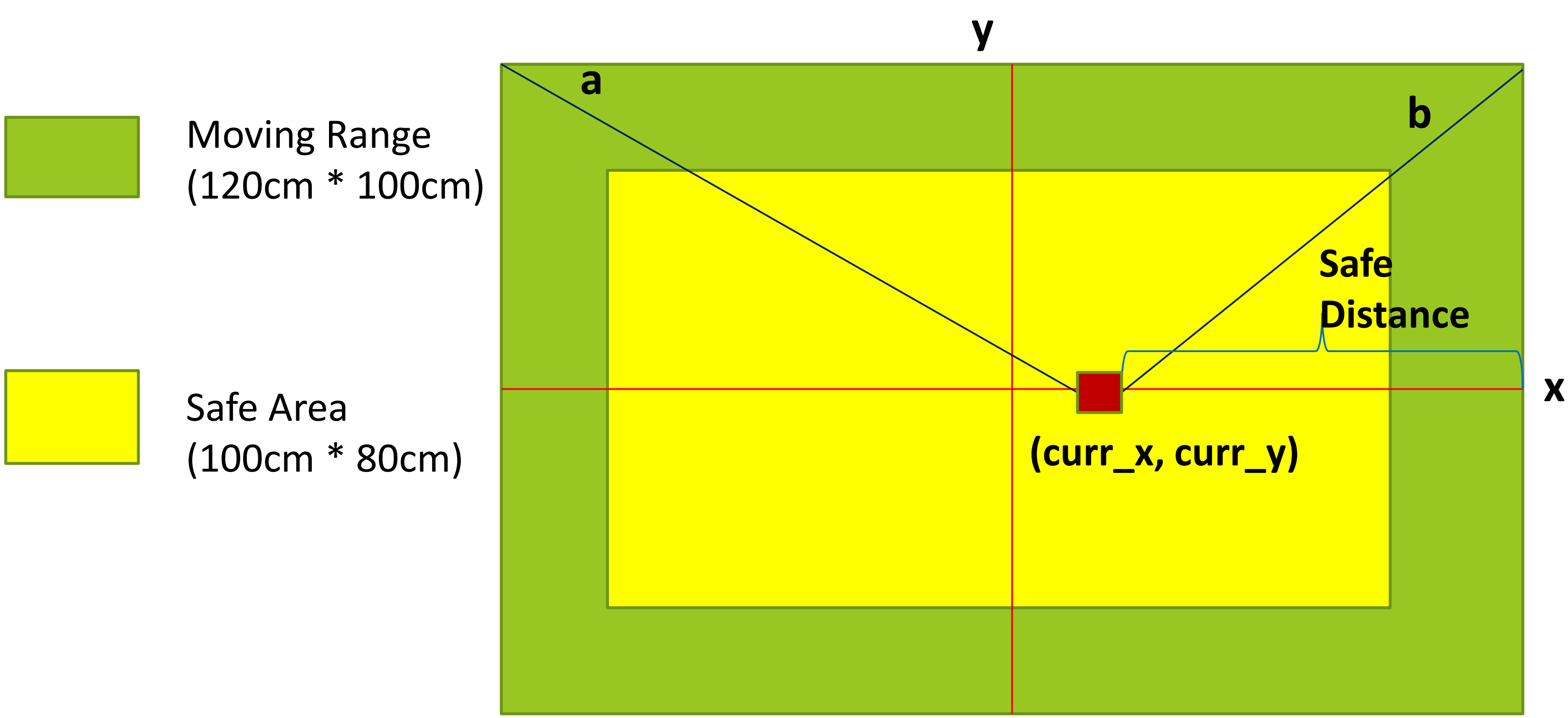
(duck moving velocity) could be very high. (More playable)

Design:
Software Diagram



Design: Calculation

Changed Variables: direction, frequency
Current Position: curr_x, curr_y



Design: Challenges

1. Deciding on the “best way” to implement the IR game aspect
2. Finding useable parts:

   - IR receivers need to have a significant range and must be feasible to connect to, only the most common frequency modulation (38kHz) is available on break out boards.
3. Verifying that the “duck” is actually in the safe communication area

Questions?