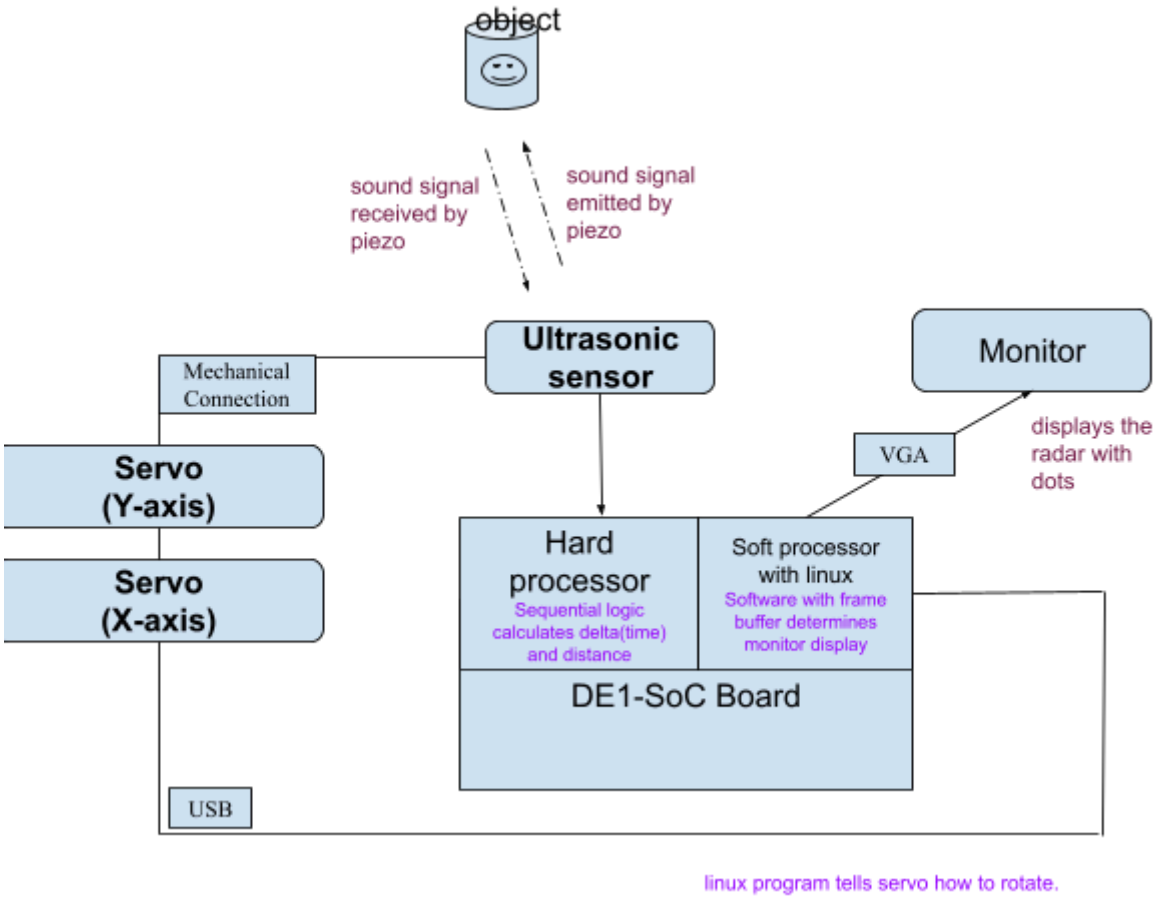
Bat Machine



Hardware components:

The following devices are connected to the FPGA:

- VGA Monitor
- HC-SR04 Ultrasonic Sensor
- Servo Motor SG90

The following mechanical connections are made:

- Ultrasonic Sensor & Servo Motor SG90
- Ultrasonic Sensor & Parabolic Reflector

Additionally, the Servo may be connected to a battery as its power source instead of the FPGA.

VGA Monitor/Graphics Rendering:

The FPGA renders the following images on the VGA monitor:

- Radar rings:
    - they can be constant, or will be flashing depending on the clock cycle (we will keep it simple)
        - to flash: keep a value that represents the radius of a concentric circle. this value goes from 0 to max and then resets to 0. the brightness of a circle (the amount of white) is a function of distance between the circle/ring, and this value.
- Spots (representing object)
    - that refresh on the radar rings:
    - can have the following organization: conceptually, we maintain a 2-d array/grid of dots (x,y coordinate pairs), evenly spaced apart by distance 2r, that each represent a potential object in space. if there is an object in space, then we store the x-y coordinates of that object in a 2-d array. this 2-d array represents objects in space. if the pixels that the VGA/FPGA is rendering are contained within r distance of a coordinate pair/element in that 2-d array, then we render those pixels in a different color, showing up to create a dot, visually representing an object in space.

HC-SR04 Ultrasonic Sensor details:

- 4 pins:
    - VCC
        - power supply, 5V
        - connected via jumper cables to GPIO
        - energy flows from the 3.3V GPIO to the ultrasonic sensor
    - Trig
        - give the Trigger to the Signal Transmitter to emit a signal
    - Echo
        - the length of time for which it goes up represents the time/distance it took for object to echo
    - Gnd
        - energy flows back from the ultrasonic sensor to the ground pin on the gpio
- Mechanical connection: servo motor via tape
- Sound wavelength: λ = Speed / Frequency
    - frequency; **HC-SR04** works at around **40 kHz**
    - speed of sound / frequency of HCSR04:
    - 340m/s / 40,000 Hz = 0.0085 m is the wavelength of the ultrasonic wave. = 8.5mm

Parabolic Reflector:

- Will be 3-D printed plastic: https://www.thingiverse.com/thing:2751650:
- width of parabolic dish should be 5-10x bigger than the wavelength of what it's reflecting: 8.5mm * 7 = 5.95cm

Servo SG90:

- Connection between the servo + the sensor mechanically must be very STRONG.
    - can begin with paper clips, then explore harder plastic materials in the makerspace
- Overview:
    - Servo can rotate approximately180 degrees (90 in each direction)
    - length of pulse corresponds to what direction to turn to/what position.
    - requires 4.8-6V of power
- 3 pins:
    - PWM: from GPIO to servo
        - pulses for 1.5ms: return to position 0. (middle)
        - pulses for 2ms: goes all the way to the right
        - pulses for 1ms: goes all the way to the left
    - VCC: possibly from GPIO voltage of 5V, or battery
    - Ground: charge flows from servo back to the GPIO ground pins
- if servo causes power surges (Specifically, if we connect the servo to the fpga also as its power source + we realize every time the servo turns on or moves the servo sucks up the current/electricity from the fpga and the fpga turns off or flickers or otherwise falters, like how turning on a fridge causes kitchen lights to flicker)
- make sure all the grounds are connected back to the same source
    - Note: since you have one GPIO ground (or 2, but of different voltages) in the FPGA, but 2 things we must connect to ground, we may want to use a breadboard to connect the grounds.
- connect the servo VCC to the battery positive or USB power supply.

Contraints file:
.qsf
set_property PACKAGE_PIN A1 [get_ports trig]
set_property IOSTANDARD LVCMOS33 [get_ports trig]

turning sound into a laser:https://www.youtube.com/watch?v=aBdVfUnS-pM

Software-Hardware interface

Sensor:
SW to HW communication:

- “please chirp/make a sound”
- “the timeout limit is ___”

HW to SW communication:

- “it took ___ time to hear echo”
- “i am still waiting for an echo”
- “your last chirp timed out”

Sensor interface in System Verilog:

input/ouput	name		meaning
input	chirp	read addr[0], [0:1]	HW: do I chirp or not?
input	timeout	read addr[1], [0:15]	HW: how long do i wait before a timeout?
output	status	write addr[2], [0:15]	HW: 0000 0000 0000 0000 : waiting for echo 1111 1111 1111 1111 : timed out all other numbers: length of time

Servo:
SW to HW communication:

Software Duties/Components:

Time → distance conversion

- There will be a module in the hardware that senses when Echo signal goes up, and begins counting the number of clock cycles. Once echo goes down (after already having come up), we take this number, divide it by the Clock frequency, to give us the time in seconds. Then, we multiply this number by the speed of sound. That will be our distance.

Retry-Mechanism:

- experiment with the time between echo + trigger again

To-do:

1. figure out the time units. AKA, what are the units for the variable ‘status’?

2. Servo interface:
    a. 16-bit value or something
        i. 0-1ms
        ii. 1-2ms
    b. Software writes a number to a register, and that changes the width of the next pulse
3. graphics

With Sonic Radar:

- https://digitalsystemdesign.in/interfacing-ultrasonic-sensor-with-fpga/ ←this has code !!
- https://youtu.be/QsPTt-7-NVg
- https://www.instructables.com/How-to-Use-Ultrasonc-Sensor-Using-Arduino/
- HC-SR04 sensor has 4 pins. Vcc, Trig, Echo and Gnd
- https://www.hackster.io/beste-caferoglu/final-ultrasonic-radar-implementation-on-fpga-fc16c3
-