Automated Aquarium System

Team #: 6752
Team Members: Dara Ros
Mandar Dessai
Nikhila Kalidindi
Srinivas Satyanarayan

Abstract: The Automated Aquarium is a system that uses electronic devices to constantly monitor and control the key environmental parameters of an aquarium. Automation helps maintain a consistent aquatic environment for flora and fauna. The first sub-system is the temperature maintenance system, the temperature sensor constantly reads the water temperature making sure that it is kept at an optimum level by using the heating and the cooling systems depending on the environment. The second sub-system is the aquarium light control system that not only makes sure that we save energy by turning the lighting system off when there is no one around but also gives the aquatic life their share of night like conditions. This system is designed in such a way that it turns the lighting system on depending on the value of the motion sensor, that is if there is moment detected outside the aquarium environment than the lights are turned ON otherwise they are turned OFF. The third and one of the most important subsystem is the air circulation system. The water in the aquarium is supposed to get muddy and dirty after specific amount of time as the water is not connected to running water source, thus an arrangement is made to flush the water out and intake fresh water after a specified amount of time so that the hygiene inside the aquarium is maintained for sustaining life in it.

Use Cases & Tests:

Use Case Name: Automated Aquarium System

Participator: User, System Admin, Motion Sensor, Temperature Sensor, Fan Based Air Circulation

Flow of Events:

- User's motion is detected and thus the light is turned ON
- Temperature sensor detects the temperature of the
- Fan for periodic oxygen/air circulation
- System admin checks for any errors in the system periodically

Entry Conditions:

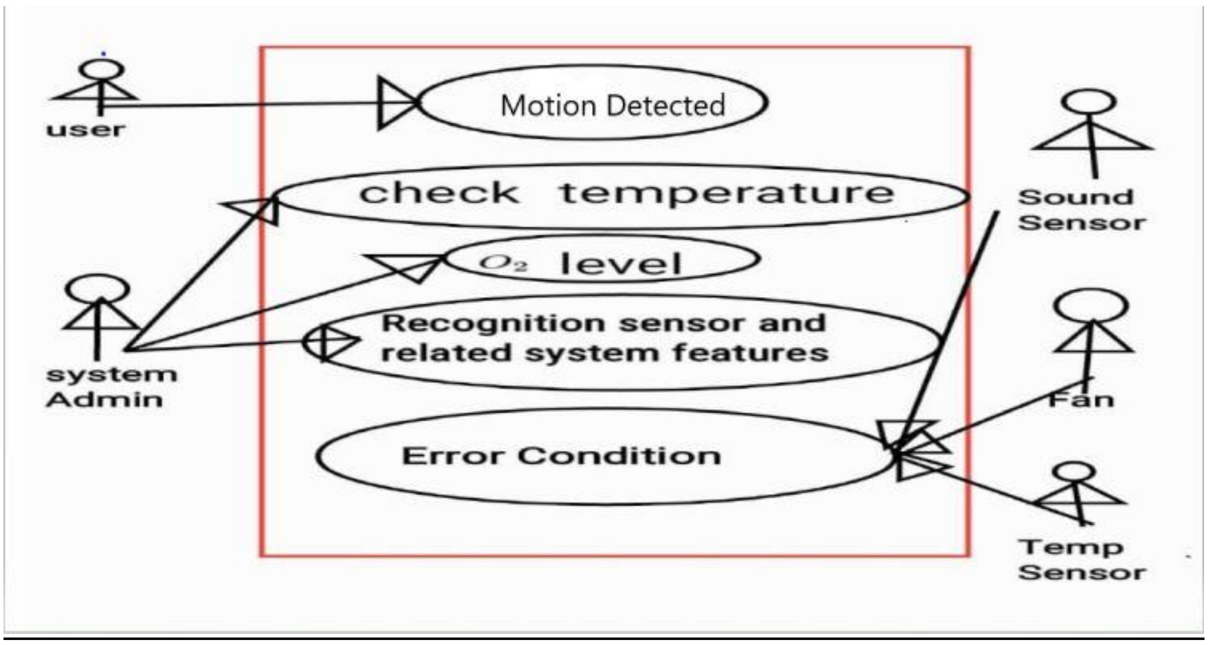
- Users motion around aquarium is detected
- Temperature drops below the minimum threshold
- Temperature rises above the maximum threshold
- Timer goes off thus indicating the start of air circulation

Exit Conditions:

- Temperature value is within the optimum threshold

- No motion is detected outside the aquarium
- The timer is running and has not gone off

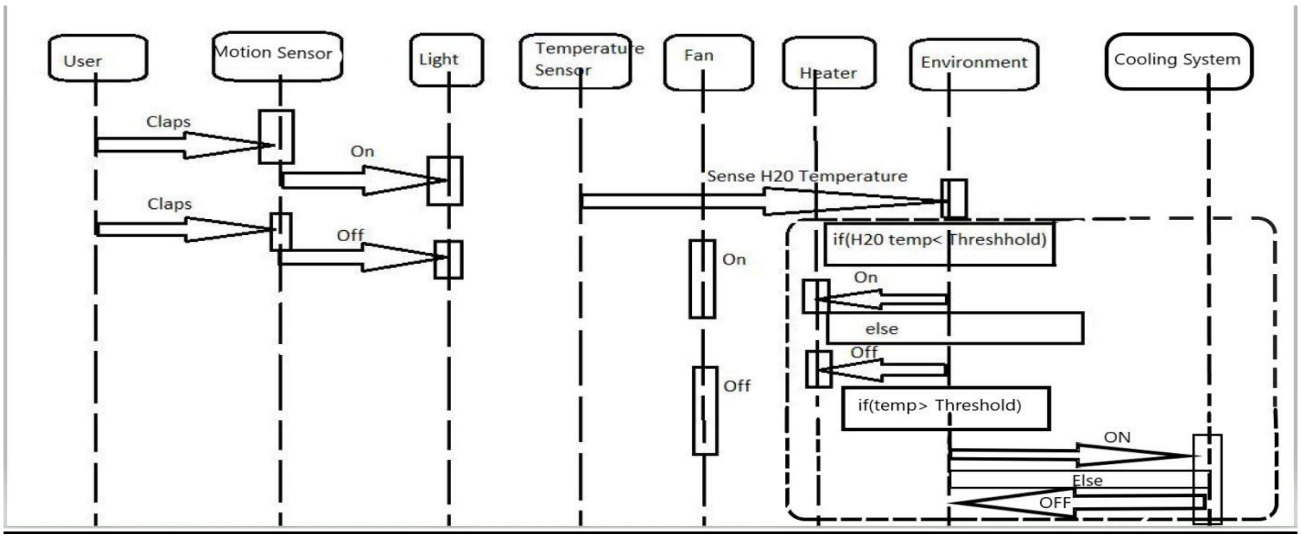
Use Case Diagram:

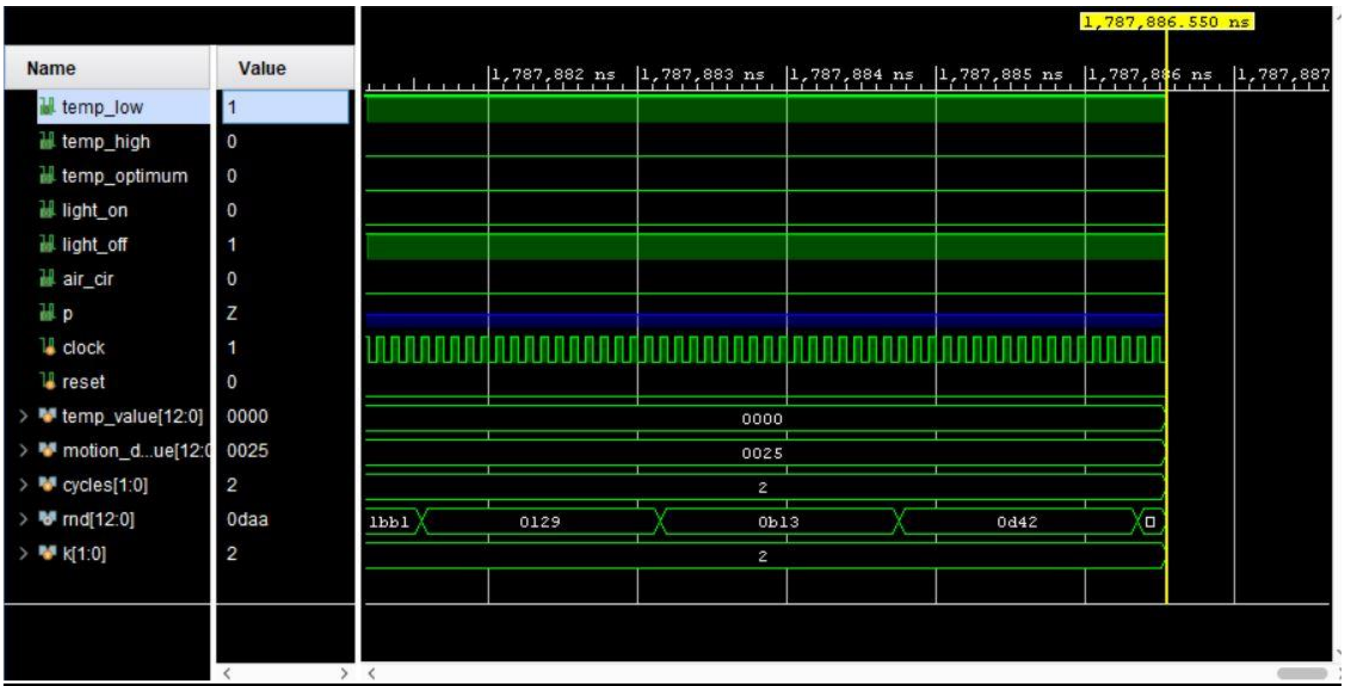


Test Results:

- The system managed to turn the output pin U16 on the Basys 3 board high in case it detected that the temperature was below the minimum threshold value (8°C)**,** thus turning on the heating system.
- The system managed to turn the output pin E19 on the Basys 3 board high in case the temperature crossed the maximum set threshold of (18°C), thus turning on the cooling system.
- The system turned on the air circulation system ON every 35 seconds.

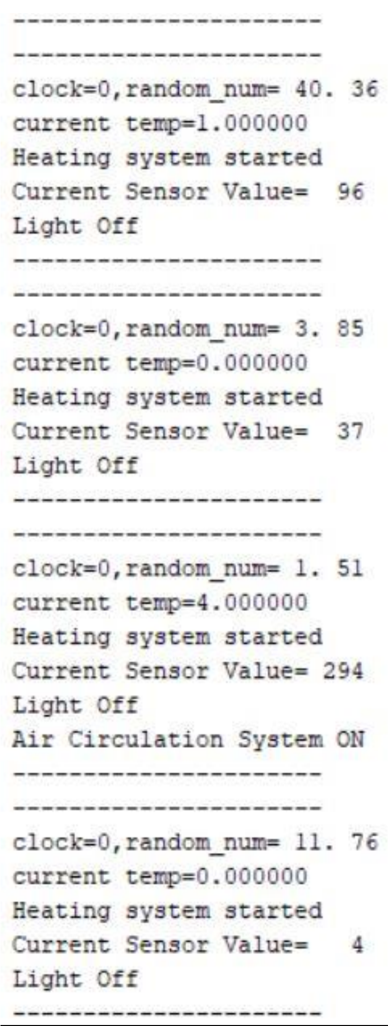
Sequence Diagrams/ Simulation Outputs





Name	Value
temp_low	1
temp_high	0
temp_optimum	0
light_on	0
light_off	1
air_cir	0
p	Z
clock	1
reset	0
temp_value[12:0]	0000
motion_d...ue[12:0	0025
cycles[1:0]	2
rnd[12:0]	0daa
k[1:0]	2

Actual Simulation Outputs



```
-----------------------
-----------------------
clock=0,random_num= 40. 36
current temp=1.000000
Heating system started
Current Sensor Value=  96
Light Off
-----------------------
-----------------------
clock=0,random_num= 3. 85
current temp=0.000000
Heating system started
Current Sensor Value=  37
Light Off
-----------------------
-----------------------
clock=0,random_num= 1. 51
current temp=4.000000
Heating system started
Current Sensor Value= 294
Light Off
Air Circulation System ON
-----------------------
-----------------------
clock=0,random_num= 11. 76
current temp=0.000000
Heating system started
Current Sensor Value=   4
Light Off
-----------------------
```

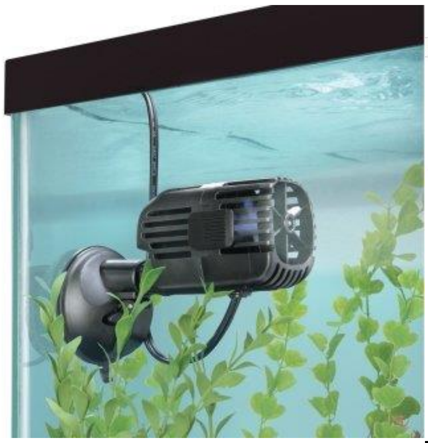
Outputs of the Corresponding Sensors in Real Time

Component Diagram and Black Box Tests

Black Box Tests:

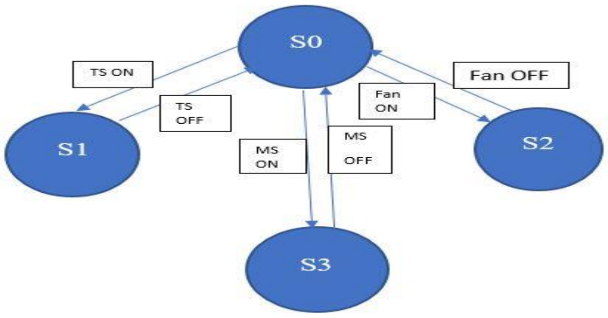
- **Motion Sensor**: Constantly checks for the motion, if the motion is detected than the lighting system is turned on by giving a HIGH on pin V19 of Basys 3 Board. The thresholds of the sensor are set such that if the sensor value is < 100 it is a logic low, that is no motion detected and if the value is >100, motion is detected. The system was designed to work with PIR motion sensor.
- **Temperature Sensor**: Constantly monitors the temperature inside the aquarium and accordingly controls the heating and cooling systems which run on pins U16 and E19 of the Basys 3 board. (Hardware assumed for project development was temperature sensor DS18B20)
- **Air Circulation System Fan**: The fan is turned ON after every 35 seconds for air circulation within and water circulation within the aquarium.

PIR Motion Sensor


Air Circulation Fan

DS18B20 Temperature Sensor

State Diagram:



TS: Temperature Sensor

MS: Motion Sensor

S0 to S1:The temperature sensor checks for the temperature in the surrounding environment and if it out of the threshold limits the heater/ cooler system is turned on.

S1 to S0: When the temperature sensor is OFF

S0 to S2: When a motion is detected, LIGHT ON

S2 to S0: When no motion is detected, LIGHT OFF

S0 to S3: Fan ON (Timer)

S3 to S0: Fan OFF(Timer)

Project Summary & Lessons Learnt

- System development and sequencing the tasks so that they can be efficient was a challenge.
- Development of code for efficient sequencing without using excessive variables and pins in Verilog was a challenge. To deal with this we had multiple iterations of test runs and tried our level best to eliminate any unnecessary redundancy of variables in the code.
- Figuring out the program for timer development in Verilog was a challenge initially, but later after multiple test runs we were able to develop and timer system for air circulation system. After extensive trial runs and online researching we were able to develop the code best fit for our system.
- Surveying out the hardware so that we could use it in our system and developing the Verilog code for these was a challenge.

Team Member Contributions

Name	**Idea**	**Logic**	**Team meet-ups**	**Initial Proposal**	**Final Report**	**Final Code**
Dara	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**Yes**
Mandar	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Nikhila	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**No**
Satyanarayan	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**No**

Difference if the project is remade:

- Invest in high quality hardware to make the system more efficient.
- Switch to a more dependable system by using industrial grade hardware.
- Create an efficient fail-safe system just in case the system fails.
- Switching to using external dependable timers for monitoring the time instead of using program based timers because there is always a possibility of hardware reset that could reset the timer.

Final Project Code

Module 1

```
module test(output temp_low, output temp_high, output temp_optimum, output  light_on,output
light_off, output air_cir,output wire p);

 // Inputs
 reg clock;
 reg reset;
 reg [12:0]temp_value;
 reg [12:0]motion_det_value;
 reg [1:0]cycles=00;
 // Outputs
 wire [12:0] rnd;
//  reg temp_low;
//  reg temp_high;
 reg [1:00]k=00;

 reg temp_low;
 reg temp_high;
 reg temp_optimum;
 reg light_on;
 reg light_off;
 reg air_cir;
 wire p;

 // Instantiate the Unit Under Test (UUT)
 LFSR uut (
  .clock(clock),
  .reset(reset),
  .rnd(rnd)
 );

 initial begin
  clock = 0;
  forever
//  #50clock = ~clock;
    #50clock=~clock;
  end

initial begin
  // Initialize Inputs
     reset = 0;
  // Wait 100 ns for global reset to finish
//  #100;
    #100
```

```
      reset = 1;
//  #200;

  #200
  reset = 0;
  // Add stimulus here
end

  initial begin

//#100 $display("clock rnd");
//$display("clock=%b,random_num=%f", clock, rnd);
end



//This is the temperature control system
always@(*)
begin
 $display("----------------------");
 $display("----------------------");
 $display("clock=%b,random_num= %.0f.%3d", clock, (rnd/100), rnd%100);
  #500000000 temp_value =rnd;
  air_cir=1'b0;
  temp_value =((temp_value)<<<2)/(1000);
//  $display("----------------------");
//  $display("----------------------");
$display("current temp=%f",temp_value);

begin
if(temp_value>=20)
begin
temp_low=1'b0;
temp_high=1'b1;
temp_optimum=1'b0;
//start cooling system
// Heating system pin= LOW
// Cooling system Pin=HIGH
k=1;
$display("Cooling system started");
end
else if(temp_value <=7)
begin
//start heating system
// Heating system pin= HIGH
// Cooling system Pin=LOW
temp_low=1'b1;
```

```
temp_high=1'b0;
temp_optimum=1'b0;
k=1;
$display("Heating system started");
end
else
begin
$display("Optimum Temperature Maintained");
temp_low=1'b0;
temp_high=1'b0;
temp_optimum=1'b1;
// Heating system pin= LOW
// Cooling system Pin=LOW
k=1;
end

end
// End Temperature control system


//start motion sensor control sys
  motion_det_value = rnd;
  motion_det_value = (motion_det_value)>>>2;
$display("Current Sensor Value=%d",motion_det_value);
if(motion_det_value>=1000)
begin
//Turn the Light LED ON

light_on=1'b1;
light_off=1'b0;
k=2;
$display("Light On");
end
else
begin
//Turn The Light LED OFF
$display("Light Off");
//stop motion sensor control system
light_on=1'b0;
light_off=1'b1;
k=2;

//assign cycles=cycles+1;
cycles=cycles+1;
end
end
```

```
always@(*)
begin
if(cycles==3)
begin
$display("Air Circulation System ON");
// Turn ON Air Circulation System LED
//assign cycles=0;
cycles=0;
air_cir=1'b1;
k=3;
end
end
endmodule
```

Module 2

```
module LFSR (
    input clock,
    input reset,
    output reg [12:0] rnd
    );

wire feedback = random[12] ^ random[3] ^ random[2] ^ random[0];

reg [12:0] random, random_next, random_done;
reg [3:0] count, count_next; //to keep track of the shifts

//always @ (posedge clock or posedge reset)
always @ ( clock or  reset)
begin
 if (reset)
 begin
  random <= 13'hF; //An LFSR cannot have an all 0 state, thus reset to FF
  count <= 0;
 end

 else
 begin
  random <= random_next;
  count <= count_next;
 end
 if (count == 13)
 begin
  count <= 0;
  random_done <= random; //assign the random number to output after 13 shifts
 end

end

always @ (*)
begin
 random_next <= random; //default state stays the same
 count_next <= count;

  random_next <= {random[11:0], feedback}; //shift left the xor'd every posedge clock
  count_next <= count + 1;
  rnd = random_done;
end


//assign rnd = random_done;
//assign rnd=rnd/100;
endmodule
```