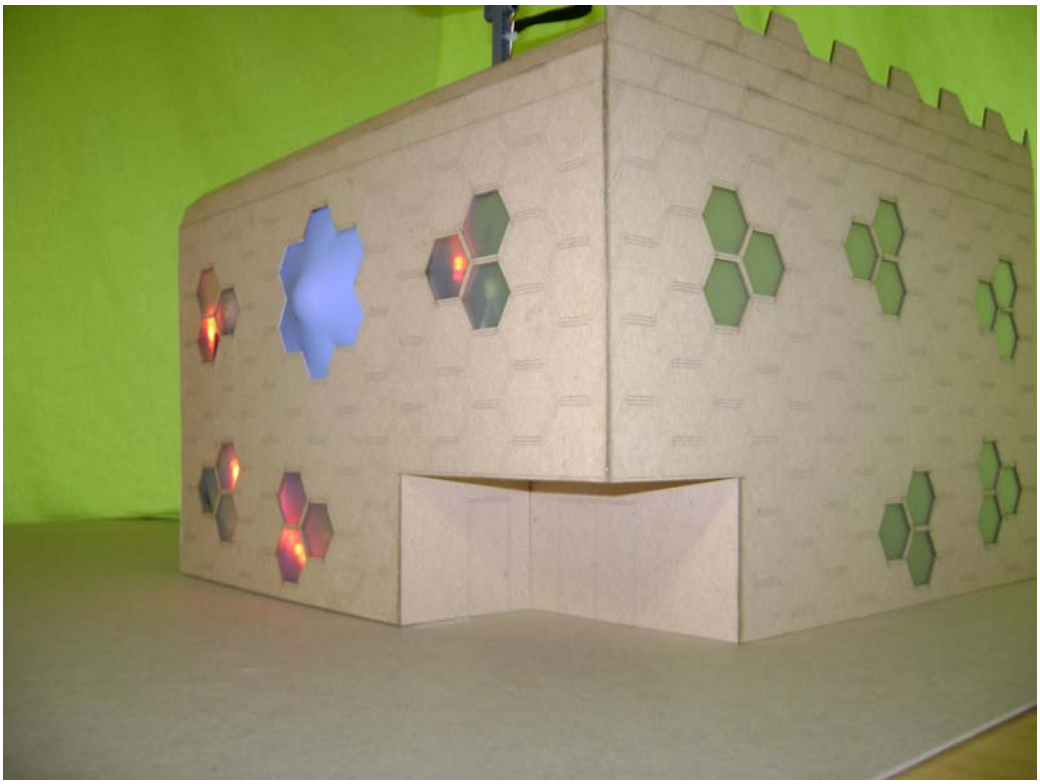
ARCH 879/ECE 893 - Architectural Robotics
Project 2 – Urban Disaster
Team members – Adam James and Paul Yanik

Shelter In a Storm



Abstract

This project presents a structure that is capable of morphing from a conventional building skin to a more aerodynamic shape that acts to dissipate the forces of high wind such as those that occur during a hurricane. A wind sensor detects the presence of wind gusts above a safe threshold. When such high wind is detected, the building self-adjusts two important components of its exterior surface: the parapet walls, and broad, flat wall surfaces. Each of these is flexed to project a curved (domed) surface toward oncoming wind. The curved surfaces act to dissipate the wind and thereby reduce the force on the underlying structure as has been shown to occur with domed buildings. After high winds have died down for a programmed period of time, the building reverts to its original, more conventional shape. Lighting is also used to outwardly express the current mode of the structure: yellow during times of normal (low) wind, and red during emergencies (high winds).

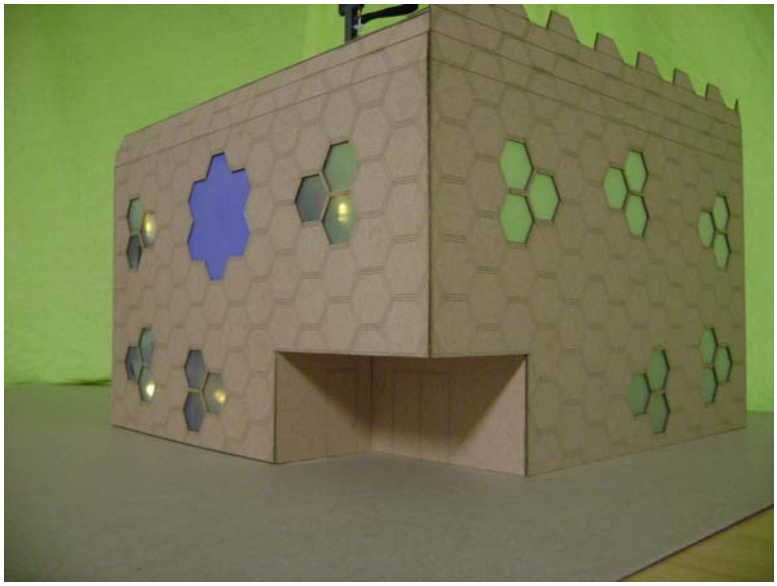
Scenario

The time before an impending hurricane making land fall in the city of Charleston, South Carolina had always been tense for its citizens. Many immediately take to the highways; seeking safety inland without knowing if they would have a home or business to return to. Many others, who would not travel due to infirmity, poverty, or misguided hubris, faced the even more worrisome specter of riding out the storm in the city.

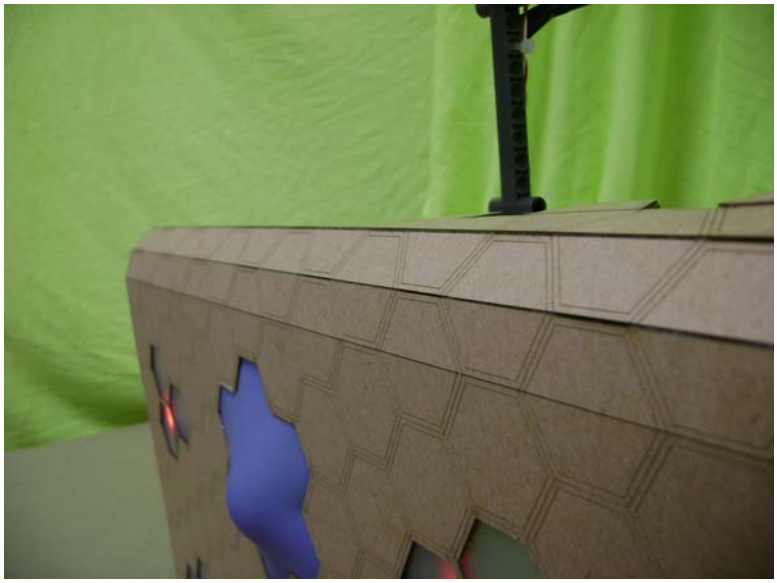
Such was the case for Rhett and Scarlett Butler. As Rhett lay sedated in his second story room in the Hollings Cancer Center at the Medical College of South Carolina, Scarlett gazed with dread out her window at the now white capped waters of Charleston Harbor. As the winds increased, she saw the running lights of the building turn from amber to red indicating emergency status. She heard the gentle hum of motors slowly arching the building's windward edges and face to deflect the fierce winds. She heard the outer panels lock into place as she watched emergency personnel guide people from the neighboring neighborhood into the building – a shelter in the storm.

Operation

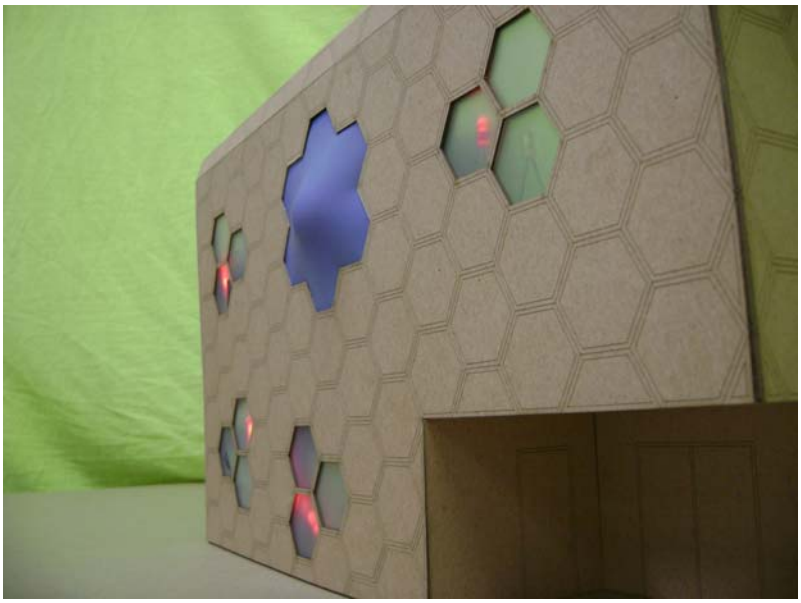
A) Lights are amber during safe winds. Aerodynamic surfaces are relaxed to a conventional shape.



C) High winds are detected. The building building's running lights change from yellow to red. Parapets morph to a aerodynamic convex shape.



C) An inflatable surface creates a convex protrusion to dissipate wind force on the flat faces of the building.



D) After winds abate, the structure returns to its original shape. The running lights turn back to amber.

List of Hardware:

- Arduino Duemilanove boards (2)
- Adafruit Motor/Stepper/Servo Shield for Arduino (1)
- Parallax servo motor (1)
- Model airplane propeller (1) (wind indicator)
- Fairchild QEC112 Infrared LED (1)
- Fairchild QSD122 Silicon Infrared Phototransistor (1) (propeller blade detector – See Figure 1)
- LEDs, Yellow (4)
- LEDs, Red (4)
- Pullup resistors (10 kΩ) (1)
- Pullup resistors (100 Ω) (9)
- Wiring
- Breadboard
- Lego Mindstorms components (for building mechanism and motor mount)
- Chipboard cutouts for outer skin
- Plastic for windows
- String

Source Code (Main Control Algorithm)

```
// ------------------------------------------------------
// Filename:      ECE983_Proj2.txt
// Course:        ECE893 - Architectural Robotics
// Team members:  Adam James, Paul Yanik
//
// Description:
// This design activates aerodyamic building surfaces
// to move into place during hurricane force winds.  The
// surfaces help to dissipate the wind energy so as to
// reduce the force experienced by the underlying structure;
// thereby reducing wind damage and debris.
// ------------------------------------------------------
#include <ServoTimer1.h>
ServoTimer1 servo1;

// ------------------------------------------------------
// Pin Assignments and Variable Declarations
// ------------------------------------------------------
// Pin assignments
int servoPin1 = 10;
int propBlade = 14;
int resetButton = 15;
int highWindPin= 16;

// Variable Declarations
int loop_count = 0;       // Count the number of loops during a time interval.
int blade_count = 0;      // Count the number of blade passes during a time interval.
int last_reading = 0;     // Last loop reading of the blade pass detector.
int curr_reading = 0;     // Current loop reading of the blade pass detector.

int TIME_INTERVAL = 300;  // Number of loops to execute while counting blade passes.
int SPEED_THOLD = 7;      // Number of blade passes needed to determine high wind.
int high_wind = 0;        // High wind indication variable.
int MAX_ANGLE = 90;       // Angle to turn servo for actuation of surfaces.
int highWindPin = 0;      // Indicator to second Arduino to actuate lighting

// ------------------------------------------------------
// IO Pin Setup
// ------------------------------------------------------
void setup()
{
  servo1.attach(servoPin1);

  pinMode(propBlade, INPUT);
  pinMode(resetButton, INPUT);
  pinMode(highWindPin, OUTPUT);

  // Open a serial link for onscreen variable monitoring.
  Serial.begin(9600);
}

// ------------------------------------------------------
// Loop for Runtime Operation
// ------------------------------------------------------
void loop()
{

  // Indicate to second Arduino to turn on yellow (LEDs) - safe mode
  digitalWrite(highWindPin, LOW);

  // Watch for high wind until it is detected
  while (high_wind == 0) {

    // Read the sensor to see if the prop blade is present.
    curr_reading = digitalRead(propBlade);
```

```
    // Display the number of blade counts (for debug).
    Serial.println(blade_count);

    // Execute the loop TIME_INTERVAL times.  Count the number of
    // propeller blade passes that are detected during the interval.
    if (loop_count < TIME_INTERVAL) {
      if ((curr_reading == 0) && (last_reading == 1)) {
        blade_count += 1;
      }
    } else {
      // If TIME_INTERVAL expires, reset the loop and blade pass counters.
      loop_count = 0;
      blade_count = 0;
    }

    // If SPEED_THOLD number of blade passes is detect during the
    // interval, assert a high wind detection.
    if (blade_count > SPEED_THOLD) {
      high_wind = 1;
      blade_count = 0;
      loop_count = 0;
      digitalWrite(highWindPin, HIGH);
    }

    // Store the current reading as the last reading to compare
    // with the blade pass detector on the next iteration.  A blade
    // pass has been detected whenever these two are different.
    last_reading = curr_reading;
    loop_count += 1;

    // Serial.println(blade_count);

  } // End while - high wind has been detected.


  // Clear the high wind indication, wait for a new detection.
  high_wind = 0;

  // Activate the motors to move aero surfaces.
  for (int r = MAX_ANGLE; r > 0; r--) {
    servo1.write(r);
    delay(40);
  }

  // Wait some amount of time before deactuating the surfaces.
  delay(5000); // 5 seconds

  // Deactuate the surfaces.
  for (int r = 0; r < MAX_ANGLE; r++) {
    servo1.write(r);
    delay(40);

  }

} // End program
```

Source Code (Lighting)

```
// ---------------------------------------------------------
// Filename:      ECE983_Proj2_lights.pde
// Course:        ECE893 - Architectural Robotics
// Team members:  Adam James, Paul Yanik
//
// Description:
// This design accommodates the sketch ECE893_Proj2.pde
// and actuages LEDs:  white when wind is low, white
// when wind is high.
// ---------------------------------------------------------
#include <ServoTimer1.h>
ServoTimer1 servo1;
```

```
// ----------------------------------------------------
// Pin Assignments and Variable Declarations
// ----------------------------------------------------
// Pin assignments
int highWindPin = 14;

int Red_ledPin1 = 13;
int Red_ledPin2 = 12;
int Red_ledPin3 = 11;
int Red_ledPin4 = 10;

int Yellow_ledPin1 = 7;
int Yellow_ledPin2 = 6;
int Yellow_ledPin3 = 5;
int Yellow_ledPin4 = 4;

int surfaces_actuated = 0;

// ----------------------------------------------------
// IO Pin Setup
// ----------------------------------------------------
void setup()
{
  pinMode(highWindPin, INPUT);

  pinMode(Red_ledPin1, OUTPUT);
  pinMode(Red_ledPin2, OUTPUT);
  pinMode(Red_ledPin3, OUTPUT);
  pinMode(Red_ledPin4, OUTPUT);

  pinMode(Yellow_ledPin1, OUTPUT);
  pinMode(Yellow_ledPin2, OUTPUT);
  pinMode(Yellow_ledPin3, OUTPUT);
  pinMode(Yellow_ledPin4, OUTPUT);

 // Open a serial link for onscreen variable monitoring.
  Serial.begin(9600);
}

// ----------------------------------------------------
// Loop for Runtime Operation
// ----------------------------------------------------
void loop()
{
  surfaces_actuated = digitalRead(highWindPin);

  Serial.println(surfaces_actuated);

  if (surfaces_actuated == 1) {
    digitalWrite(Red_ledPin1, HIGH);
    digitalWrite(Red_ledPin2, HIGH);
    digitalWrite(Red_ledPin3, HIGH);
    digitalWrite(Red_ledPin4, HIGH);
    digitalWrite(Yellow_ledPin1, LOW);
    digitalWrite(Yellow_ledPin2, LOW);
    digitalWrite(Yellow_ledPin3, LOW);
    digitalWrite(Yellow_ledPin4, LOW);
    } else {
    digitalWrite(Red_ledPin1, LOW);
    digitalWrite(Red_ledPin2, LOW);
    digitalWrite(Red_ledPin3, LOW);
    digitalWrite(Red_ledPin4, LOW);
    digitalWrite(Yellow_ledPin1, HIGH);
    digitalWrite(Yellow_ledPin2, HIGH);
    digitalWrite(Yellow_ledPin3, HIGH);
    digitalWrite(Yellow_ledPin4, HIGH);
  }

} // end
```

Discussion:

Examples in literature and qualitative simulations of curved structures under high wind show that these shapes effectively help dissipate the force of wind. Further, the structure need not be fully domed to take advantage of these benefits. Rounded surfaces on the crisp corners and flat surfaces of a building help to accomplish the same goals. Our building shows key instances of where this paradigm can be implemented: on the parapet walls and flat faces of a building. Given that one instance of our was an inflatable surface, an area of future work would be to explore possible materials for implementation of an inflatable surface.

Problems Encountered

A wind speed indicator was improvised using a propeller blade with sensors that detected the passing of a blade past a fixed point. The number of passes was counted to determine the spinning speed of the blade (which was used to reflect wind speed). The algorithm used to count blade passes was an effective and could be used generally as a sampling algorithm for any repetitive event.

A second Arduino board was used to control the lighting of the structure. An output from the first board was used as a signal for the lighting to change states. In order to accomplish this, it was necessary to unify the grounds between the two boards.

Future work on the general concept would include creation of curved surfaces that enclosed building corners at parapet walls. This shape posed a mechanical design challenge that was not addressed on this project. Also, the material that would be used to enclose the inflatable surface represents an unknown technology. Possible solutions include Kevlar fabric inflated with recyclable beads to produce a semi-rigid curved body. This concept could be extended to address the problem at building corners mentioned above.