Project Title: **Automatic Storm Shutters**

Team Name: **Make It Rain**

**Team Members:**
Name: Kyle Weber  
Name: Zachary Wernlund

**Project Abstract:**

Automatic storm shutters will close in the case of strong winds, essentially a large scale storm. We will also include a weather station to record various weather aspects. Through a serial connection, a computer will update Twitter with the weather conditions as well as the shutter status, so that the user can access this information at any time from any location with internet access.
Team: Make It Rain

Table of Contents:

Product Features/Objectives Page 2
Analysis of Competition Page 5
Bill of Materials Page 6
Concept Selection Page 8
Project Architecture Page 9
Division of Labor and Gantt Chart Pages 10-11
References and Appendices Page 12

List of Figures and Tables:

Figure 1: Example Anemometer Page 2
Figure 2: Weather Station Page 3
Figure 3: Storm Shutters Page 4
Figure 4: Block Diagram Page 5
Figure 5: Detailed Block Diagram Page 9
Figure 6: Division of Labor Page 10
Figure 7: Gantt Chart Page 11
Team: Make It Rain

Product Objectives and Features:

The weather station used to implement this product includes wind speed, wind direction, and rain fall (in inches). A separate sensor will be added to record temperature as well as the previously mentioned weather aspects. The most important aspect is the wind speed, which will be used to detect dangerously high wind speeds. Under this condition the storm shutters will automatically close.

The weather station’s anemometer is operated using a reed switch, which is basically a switch that opens and closes when necessary. The period of the wind speed is calculated and used to find the frequency. From that frequency, there is a value (1.423 Hz) to be multiplied by in which to convert to miles per hour. The wind vane uses a variable resistor which changes resistance based on the direction it is pointing. Simply using an analog to digital converter we were able to give each direction a digital value. We then use a look-up table to determine the direction, out of sixteen possible directions (N, NE,
Team: Make It Rain

NNE, etc.). The rain fall system also implements a reed switch, which tilts side to side and counts the number of “dumps.” Each time the system tilts is equal to 0.011 inch of rainfall. The last sensor used is the temperature sensor, which also varies resistance based on the temperature level. This is accomplished using an analog to digital converter as well.

![Figure 2: Weather Station](image)

After the wind has reached a certain speed for a certain amount of time, a signal will be sent to the control circuit inside the home, letting the motor know to shut the shutter. The shutters will then lock in place until the user manually opens them, through a push button. The locking feature allows for a security measure, since most homes are subject to theft and destruction of property after storms.

The automatic storm shutter system can be used in any home that risks the chance of destruction during a storm. If the resident is out of town or is not at their vacation home, the storm shutter system
will close and protect their home without worry. This would be applicable for any hurricane, tornado, tropical depression, tropical storm, etc. The following image shows the type of storm shutters we would eventually use in our design.

![Example Storm Shutters](image)

**Figure 3: Example Storm Shutters**

The last important feature of the product is the ability to check the weather conditions and the status of the storm shutters at any location. For this, a serial connection to a CPU was implemented. Through this serial connection one can use code to access Twitter and update the weather conditions and status at any time. This will update automatically as long as the system is up and running. If the system goes down, Twitter will have to be reconfigured and set up again.
Analysis of Competitive Products

Research has shown that there is no real competition as for our design as a whole. There is a patent out for a wired or wireless control of automatically lowering storm shutters in the presence of rain and we found complex systems that track a variety of weather conditions. Our idea is novel in the sense that we plan to use a several sensors to determine dangerous conditions and the system will respond accordingly.

Each of the above mentioned systems function independently and are used for a singular purpose. Our idea integrates a weather station, wireless communication to the storm shutters and the ability to access weather data at your home at any time by use of the internet.

We feel that rain does not necessarily mean storm shutters should close. The true damage occurs under heavy winds. Different developers have run into a variety of problems when creating their automatic storm shutters that close due to heavy rain. We plan to bypass all of the rain problems they encounter by sensing wind speed rather than rain.
Team: Make It Rain

Technical Objectives

The automatic storm shutters will use an anemometer detection system using a sensor that is sensitive enough to determine the number of rotations in a certain amount of time. This will determine when wind speeds have reached around 60 to 70 mph, possibly less. Once that has occurred we will use an XBee to transmit a wireless signal to the home control system. This will use an Atmel microcontroller. Once at the home control center another Atmel microcontroller will control the manual pushbutton for close and open. This is where the system will determine whether or not to close the shutters, open the shutters, or do nothing. The shutters will run off of a motor of choice and lock when in the down position. Power consumption should not be too much of an issue. The following block diagram best represents our technical objectives in this project.

![Block Diagram]

Figure 3: Block Diagram
Team: Make It Rain

Primary Goals:

- Read and track high wind speed and use that data to control storm shutter
- Accurately track, record, and process wind speeds through use of Atmel processors
- Create a switching system that will be able to control storm shutters (or a similarly functioning device)
- Have the two systems communicate with each other
- Ensure the wind speed is read accurately and shutter system operates under extreme conditions

Secondary Goals:

- Track wind direction, temperature, and rainfall
- Have the two systems communicate with each other wirelessly

Bill of Materials

Weather Station (anemometer, wind vane, rain gauge) $70

L235 Temperature sensor $10

(2) Atmel ATmega 32 $10

(2) Xbee Wireless Communication chips $40

5V DC Motor and Gear box $10

FTDI Serial to USB communication device $10

Twitter account (if you have internet) Free

Miscellaneous materials (wood, parts, cables, etc.) $60

Total Cost (approximately): $210

Mechanical, automatic operated storm shutters can run for $1000 or more; our system will not use real storm shutters but a mock up to demonstrate the functionality of the system.
The key selection choice of project was the decision between a wired system and going wireless. Our design makes use of XBee transmitters for wireless communication. Wireless presents several advantages of using a wired design. Our system is focuses on emergency response, it would seem that a wired system would be ideal because it would be a fail safe in the event of power failure. Since our system is meant to pre-empt severe weather and wireless communication is a reliable source, wireless is an effective way to communicate with our shutters. The wireless system allows for the user to easily add more shutters with out the hassle of running new cable to install them; just as easily they can be uninstalled.
Team: Make It Rain

Project Architecture

Figure 5: Project Flow Chart
Team: Make It Rain

Division of Labor and Gantt Chart

For the majority of the project, team members will share responsibilities because each part requires a certain amount of analog and digital portions.

<table>
<thead>
<tr>
<th>Task</th>
<th>Kyle Weber</th>
<th>Zac Wernlund</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Research/Project Proposal</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Part Ordering</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Weather Station</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>XBee Communication</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Shutter/Motor Control</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Twitter Updates</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Testing</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Report</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Demo</td>
<td>50%</td>
<td>50%</td>
</tr>
</tbody>
</table>

Figure 6: Division of Labor
Figure 7: Gantt Chart
References and Appendices

First anemometer (Figure 1): http://monterinesf1.blogspot.com/2008/10/first-anemometer.html


Shutter Image (Figure 2): http://www.fema.gov/mitigationbp/fileDetails.do?id=2568

Appendices:

WEATHER STATION

SIGNAL(SIG_INTERRUPT0)
{
    //gauge tripped, increase global count value
    rainCount++;
    _delay_ms(1);
}

SIGNAL(SIG_INTERRUPT1)
{
    //pushbutton reset
    rainCount = 0;
}

unsigned int windspeed(void)
{
}
unsigned int WIND_PIN_IN5 = (PINA & 0b00000010); // Sets Pin 0 of C to be read

    long unsigned int ticks = 0;

long unsigned int i=0;

unsigned int j=0;

long unsigned int sum;

    unsigned int windspeed;

float avg, freq;

int wind[10];

while (j <= 11)
{

    while(WIND_PIN_IN5 != 0)        // Loop to count time while Pin5 is high
    {

        WIND_PIN_IN5 = (PINA & 0b00000010);

        _delay_ms(1);

        i++;

        if (i >= 2500)
            break;

    }

    while(WIND_PIN_IN5 == 0)        // Loop to count time while Pin5 is low
    {

        WIND_PIN_IN5 = (PINA & 0b00000010);

        _delay_ms(1);

    }
Team: Make It Rain

```c
i++;

if (i >= 2500)
{
    ticks = ticks + i;
    break;
}

if (j == 0) // Ignores first data
{
    i = 0;
    ticks = 0;
    j++;
}

if (j != 0) // Store into wind array
{
    wind[j-1] = i;
    if (j != 0)
      // Ignores first data
    {
        i = 0;
        ticks = 0;
        break;
    }
}
```
i = 0;

ticks = 0;

windspeed = 1;

j++;
}
}

if (windspeed == 0)
{
    return windspeed;
}

else
{
    sum = 0;

    for (int x=0; x <= 10; x++)
    {
        sum = sum + wind[x];
    }

    avg = (sum/10.0);

    freq = 1000.0/avg; //convert from ms to s at the same time

    windspeed = freq * 1.492;
    return windspeed;
}
void shutter(int windspeed)
{
    int OPEN = (PINC & (_BV(2)));
    if ((windspeed > DANGER) && (shutter_status==0))
    {
        i++;
        _delay_ms(200);
        if (i >= SUSTAINED)
        {
            UCSRB = 0b00000000;
            PORTC |= (_BV(1));
            PORTC &= ~(BV(0));
            for (int j=0; j<65; j++)
            {
                _delay_ms(200);
            }
            PORTC &= ~(BV(1));
            shutter_status = 1;
            UCSRB = 0b00011000;
        }
    }
}
if (windspeed <= DANGER) i=0;

if ((OPEN == 0) && (shutter_status == 1))
{

    UCSRB = 0b00000000;
    PORTC &= ~(BV(1));
    PORTC |= (BV(0));
    for (int j=0; j<70; j++)
    {
        _delay_ms(200);
    }
    PORTC &= ~(BV(0));
    shutter_status = 0;
    UCSRB = 0b00011000;
}

}

int main(void)
{

    Term_Erase_Screen();
    USART_Init();
    Shutter_Init_Ports();
    lcdInit();
    lcdClear();
```c
i = 0;
shutter_status = 0;

unsigned char dir,status;
long unsigned int rain_deci, rain_int, rain,rainScale;
unsigned int rainLSB, rainMSB;
unsigned int speed, analog_val, temp;

unsigned char s_speed[10], s_dir[10], s_rain[10],s_temp[10], s_status[10];
unsigned char lcd_speed[10], lcd_rain[10],lcd_temp[10];
unsigned char buf[10];

while(1)
{
    speed = RX_byte();
    analog_val = RX_byte();
    rainLSB = RX_byte();
    rainMSB = RX_byte();
    temp = RX_byte();

    rain = ((rainMSB*256)+rainLSB);
    rainScale = rain*11;
```
Team: Make It Rain

rainScale = rainScale/2;

rain_int = rainScale / 1000;

rain_deci = rainScale % 1000;

dir = Wind_Table(analog_val);

shutter(speed);

//LCD Functions
itoa(speed,buf, 10);
strcpy(lcd_speed,buf);

itoa(rain_int,buf, 10);
strcpy(lcd_rain,buf);
strcat(lcd_rain,".");
itoa(rain_deci,buf, 10);

if (rain_deci < 100) strcat(lcd_rain, "0");
strcat(lcd_rain,buf);

itoa(temp,buf, 10);
strcpy(lcd_temp,buf);

//LCD Print
lcdClear();
lcdGoto(0,0);
lcdString(dir);
lcdString(" ");
lcdString(lcd_speed);
lcdString(" mph SS:");
if (shutter_status == 0) lcdString("O");
else if (shutter_status == 1) lcdString("C");
lcdGoto(1,0);
lcdString("Rain:");
lcdString(lcd_rain);
lcdChar(quote);
lcdString(" ");
lcdString(lcd_temp);
lcdChar(degree_sym);
lcdString("F");

//To Twitter functions

//Temperature
strcpy(s_temp, "t:");
itoa(temp,buf, 10);
strcat(s_temp, buf);
TX_string(s_temp);
_delay_ms(100);

// Windspeed
strcpy(s_speed, "s:");
itoa(speed, buf, 10);
strcat(s_speed, buf);

TX_string(s_speed);
_delay_ms(100);

// Wind Direction
strcpy(s_dir, "d:");
strcat(s_dir, dir);

TX_string(s_dir);
_delay_ms(100);

// Rainfall
strcpy(s_rain, "r:");
itoa(rain_int, buf, 10);
strcat(s_rain, buf);
strcat(s_rain,".");

itoa(rain_deci,buf, 10);

if (rain_deci < 100) strcat(s_rain, "0");

strcat(s_rain,buf);

TX_string(s_rain);

_delay_ms(100);

//Shutter Status

strcpy(s_status, "h: ");

if (shutter_status == 0) strcat(s_status, "O");

else if (shutter_status == 1) strcat(s_status, "C");

TX_string(s_status);

_delay_ms(100);

}

return 0;

}