Intelligent Rain Window Protector 3.1

May 8th 2017

James Wyeth, Edward Ellis
ECE 110/120 Honors Lab
Spring 2017

Photo: Miracle Forest

Introduction

Have you ever come home from a rainy day at work to find that the rain made its way into your house? Maybe you forgot your window was open, maybe you didn't know it was going to rain, or maybe your window just isn't able to close properly. This is the problem that we were trying to solve when we started this project. Even if you were at home when it happened, having water enter your house through a neglected window can be devastating. Our project is a convenient solution to this problem using simple electronics, dubbed the Intelligent Window Rain Protector 3.1 (iWINRAPR).

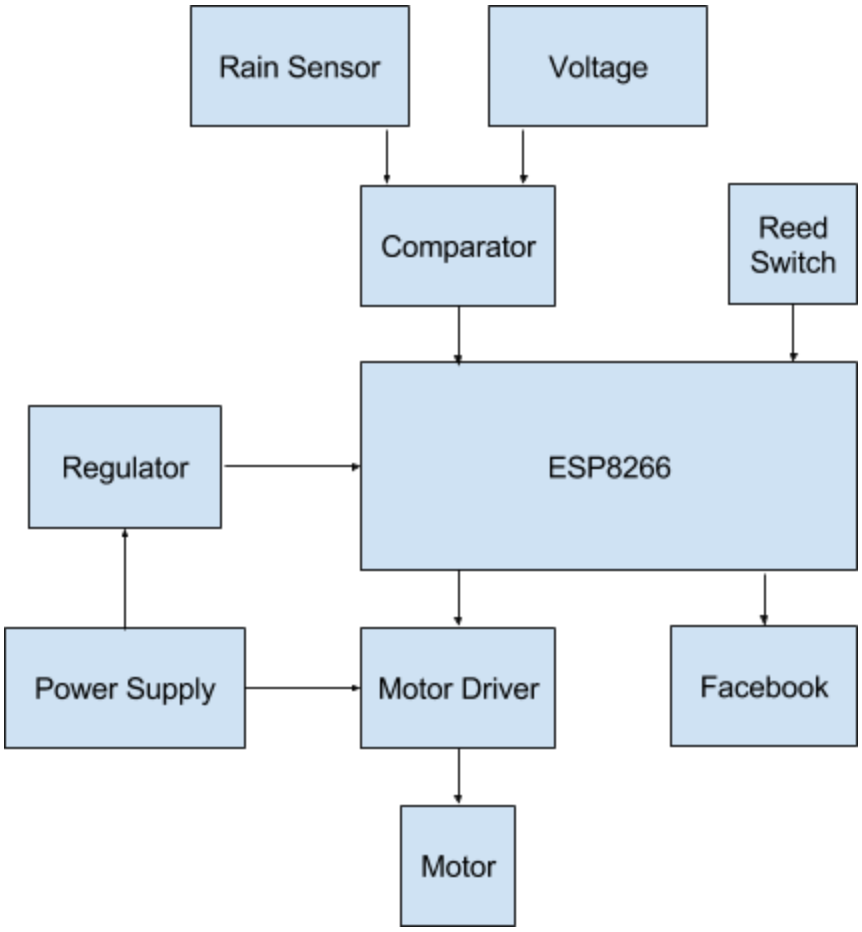
Water is a very damaging substance. It rots wood, facilitates the growth of mold, and, perhaps most importantly, it can destroy electronics. This system utilizes an analog rain sensor to decide when your window needs to be closed, protecting your house from any would-be water damage, saving you money and insurance hassle. The window is closed using a standard 12V motor, and there is a sensor to tell the system when the window is closed. The system is powered using a single 12V power brick that plugs into the wall.

Last semester we had a prototype for this project, but it wasn't near complete. It used multiple microcontrollers and it was all on a breadboard. This semester we transferred the design onto a PCB, making the system a single, small board that can fit anywhere. There were 2 iterations of the PCB, the first designed to piggyback on another board, with the second moving everything onto a single board. On the latest iterations, a power regulator was added to remove the necessity of a USB charger or battery. The board is designed to make connecting everything quick and easy, so anyone can pick one up and install it in a few minutes.

Design

The basic design of the system is shown in this block diagram. The rain sensor provides an analog signal to the comparator which compares it with a fixed voltage. The voltage was determined during testing of our system from last semester. The reed switch is a magnetic switch that determines when the window is closed. The power supply powers the motor driver directly to provide the motor power, and goes through a 3.3V regulator to power the rest of the circuit. The ESP8266 is the main controller for the system. It reads the sensors and supplies a signal to the

motor driver. It also connects to the internet and communicates with Facebook to send messages over messenger.



When any water (or anything conductive) touches the rain sensor, its resistance decreases, lowering the voltage of a voltage divider. The comparator compares this to the voltage supplied by the calibrated setpoint, and sends a digital signal to the ESP8266 letting it know whether or not it's raining. When this signal turns on, the ESP8266 checks if the window is open. When the reed switch is open, the signal it sends is low, and the window is open. When the window closes, the switch closes, sending a positive signal to the ESP8266. If the window is open, it sends a PWM signal to the motor driver, which uses an H bridge to drive the motor in the desired direction. When the window is detected as closed, the motor stops being driven. When the window is closed, the user is alerted over facebook messenger. In addition, the ESP8266 knows the states of the system at all times, and can be asked for them through Facebook Messenger. The schematics and PCB layouts can be found in Appendix A.

Results

Most of the sensor characterization and testing was performed last semester when the original prototype was being made. We switched from a 9V power supply to a 12V power supply, which made the motor a bit faster. This made the gap after closing the window 0, but this isn't necessarily an improvement, since the reed switch position needs to be set differently for each window to close it fully.

The first PCB was a success. It wasn't ready on time for the first day of EOH, but it worked perfectly on the second day. It got a lot of attention from people, and seemed to be something people would want in their houses.

Problems and Challenges

The initial PCB had a lot of problems, and it arrived less than a week before EOH. I made some modifications to it and redlined the schematic, which involves drawing lines on it in red pen. This is a procedure used in PCB prototyping to make it clear what needs to be changed on the next prototype without changing the original schematic. It also shows how the modified boards are wired when troubleshooting. I had to modify the comparator circuit and add the reed switch circuit since I had made mistakes when originally designing it. Thankfully I didn't make any mistakes with the motor driver, since it's so small they would have been impossible to fix.

Also, once EOH was over, the PCB broke and was largely forgotten for a while as other things required attention. I revisited it and redesigned the PCB twice. I sent the PCB and parts off to be ordered, and the PCB order was delayed. The parts arrived within a couple of days, but I have yet to see the PCBs. This means that the voltage regulator hasn't been tested, and it's not guaranteed to work without any modifications. Also, since I expected to have the PCB before the end of the semester, I didn't put the time in to fix the first prototype until it was too late and I ended up presenting a broken system to the professor.

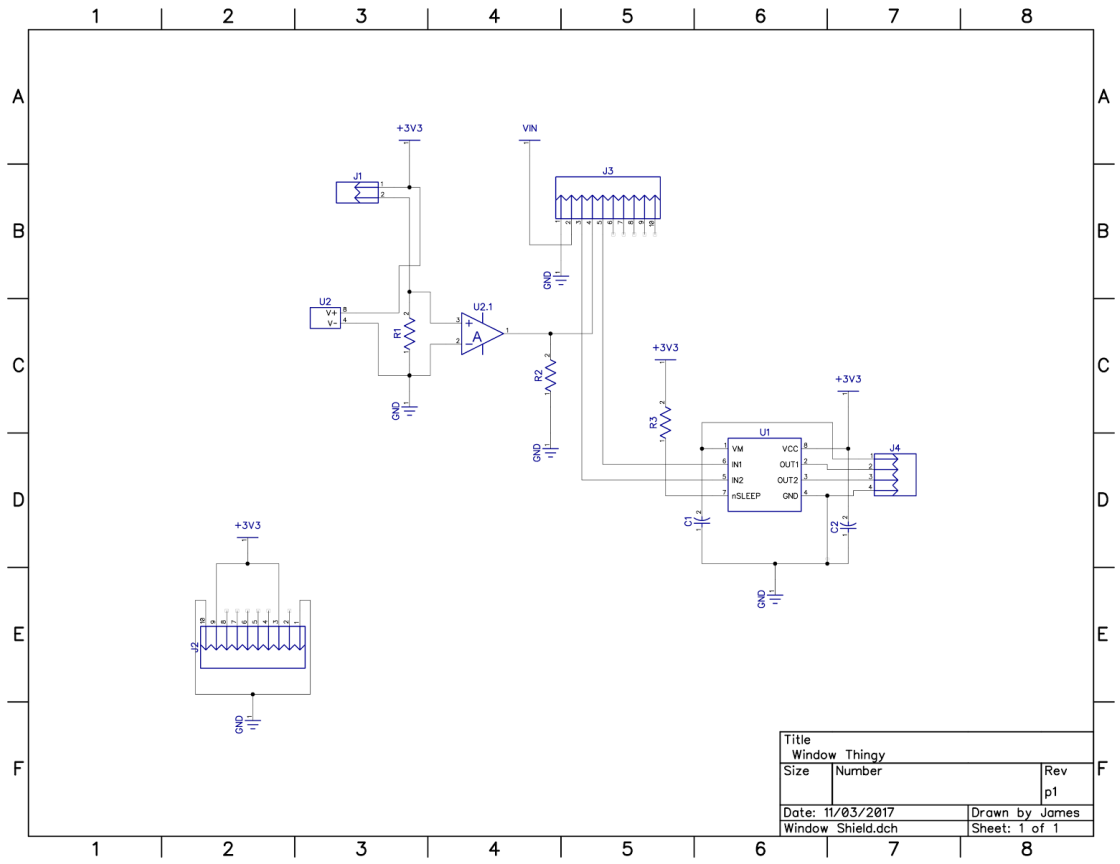
Future plans

Next semester we plan on completing the current iteration of the PCB. We will assemble and test it, and if any more issues are found, we will make the necessary modifications and order a new prototype.

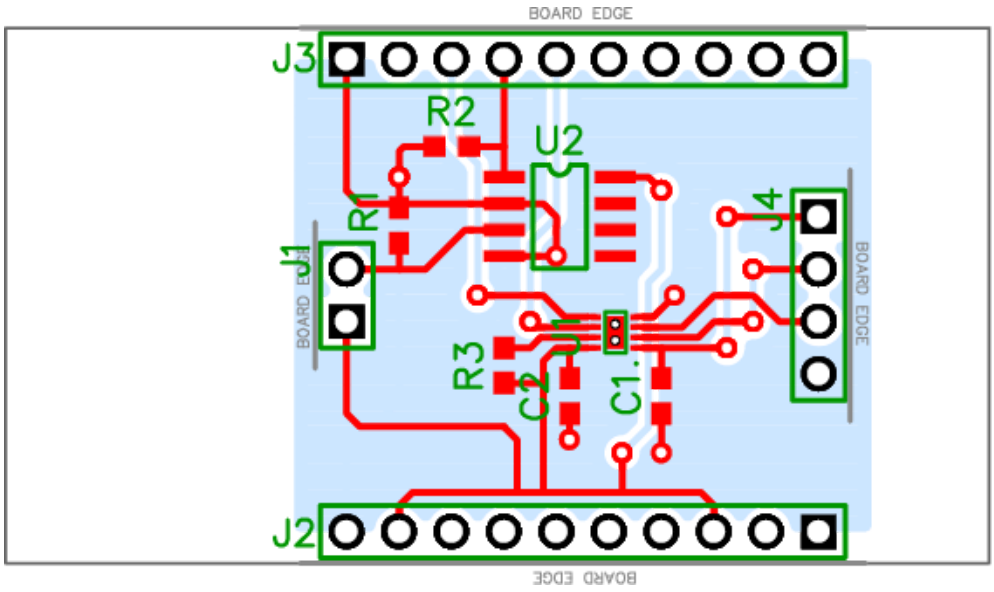
We also plan on finishing messenger integration. Currently a message is sent to the owner if the window is closed, and data can be requested all at once. We plan on adding the ability to send instructions and customize notifications. Currently the window has to be re-opened manually, and the system won't allow it to be opened until the sensor is completely dry. We plan on letting the owner open and close the window remotely, and choose between different notifications. These could include notification of an open window at a specific time each day, of a window being opened, of the window closing itself, and several others.

We also plan on making a better rain sensor. Our current system requires the sensor to be outside with a wire somehow being fed inside. Although this is quite cheap, it is inconvenient. Some cars have rain sensors that sense rain from the inside of the window, and we plan on creating a cheaper version of this to make installation more convenient and visually pleasing.

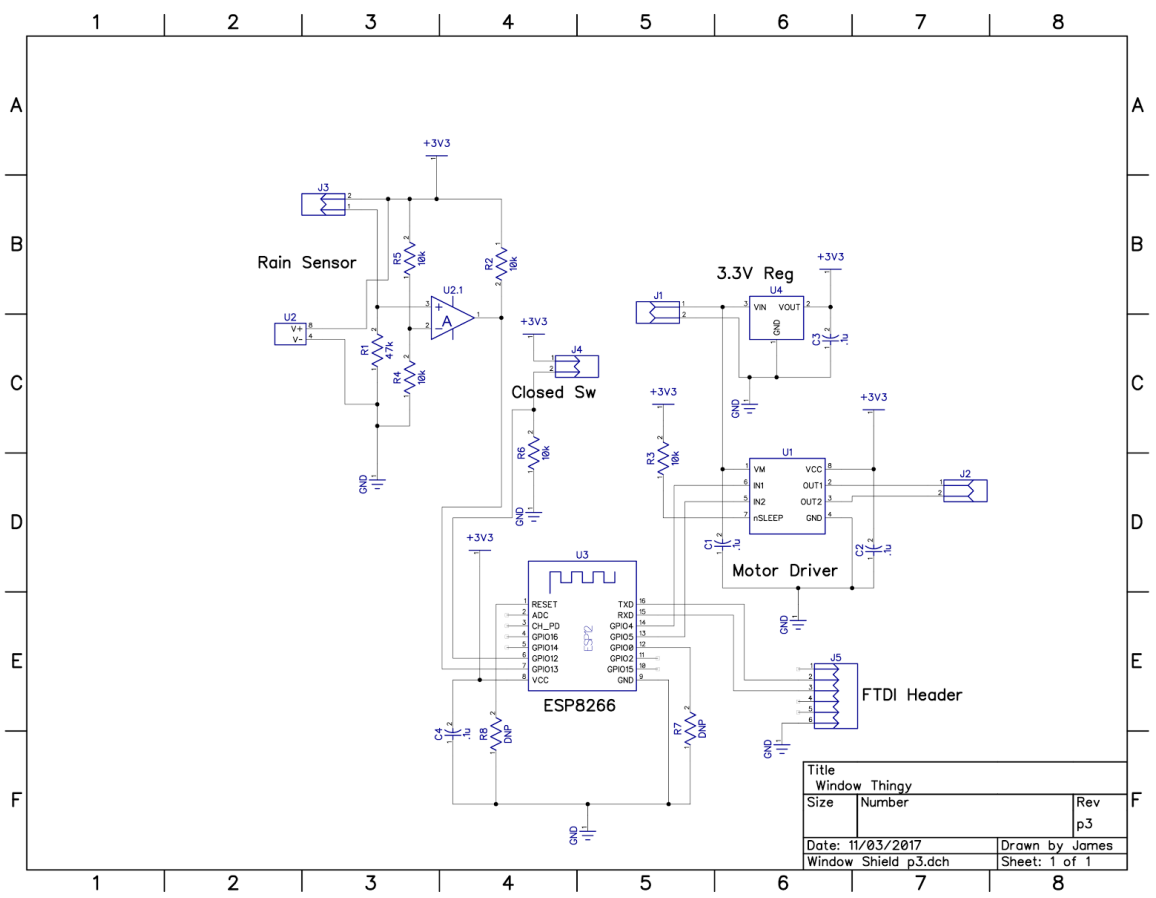
Appendix A - Design



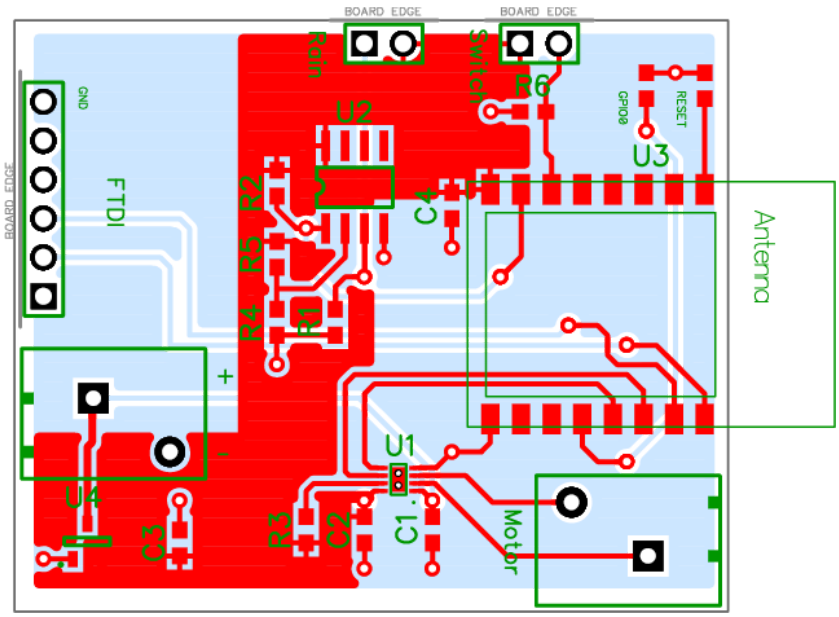
First prototype PCB schematic



First prototype PCB layout



Latest prototype PCB schematic



Latest prototype PCB layout

Appendix B - Code

```
#include <FirebaseArduino.h>
#include <ESP8266WiFi.h>

#define FIREBASE_HOST "ajario-2b883.firebaseio.com"
#define FIREBASE_AUTH "uJyP8aE33JAbONiSSW6SaEZhvPB6wLLFsWqLLqIb"
#define WIFI_SSID "IllinoisNet_Guest"
#define WIFI_PASSWORD ""

#define SLEEP_TIME 50

#define OPEN_DIR 1

int swIn = 12;
int rain = 13;
int motor1 = 4;
int motor2 = 5;

int sendCounter = 0;

void changeStatus(int newStatus) {

}

void setup() {
  pinMode(swIn, INPUT);
  pinMode(rain, INPUT);
  pinMode(motor1, OUTPUT);
  pinMode(motor2, OUTPUT);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop() {
  int raining = digitalRead(rain);
  int closed = digitalRead(swIn);

  digitalWrite(motor1, 0);
```

```
  if(!closed && raining)
  {
    analogWrite(motor2, 50);
  }
  else
  {
    analogWrite(motor2, 0);
  }

  if(sendCounter > 100){
    sendData();
    sendCounter = 0;
  }
  sendCounter ++;

  delay(10);
}

void sendData(){

  if (WiFi.status() != WL_CONNECTED) {
    return;
  }

  // set value
  Firebase.setInt("closed", closed);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /closed failed:");
      Serial.println(Firebase.error());
      return;
  }

  // set value
  Firebase.setInt("raining", raining);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /raining failed:");
      Serial.println(Firebase.error());
      return;
  }
```

```
  // append a new value to /logs
  String name = Firebase.pushInt("logs", raining);
  // handle error
  if (Firebase.failed()) {
      Serial.print("pushing /logs failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

}
```