Automated Window Temperature Regulator

Derik Lee, Hersh Singh, Louis Wright
Team 22

Final Report for ECE 445: Senior Design
Fall 2019

December 11, 2019

Abstract

This document describes the design process undergone to produce an automated window temperature regulator. The purpose of the window is to utilise the difference in climate inside and outside a building to make a decision to be open or closed. We will first introduce the problem we encountered and how we will solve the problem with our window. There will be an overview of our modules and then go into depth on each part of our system. Testing and results will then be presented at the end.

1. Introduction

1.1. Objective

Most houses and buildings tend to consume a lot of energy in the form of electricity through the HVAC (Heating, Ventilating and Cooling) system. This system is used to regulate and maintain temperature and air quality at a comfortable level for the occupants. As the threat of climate change grows, new technology is emerging to curb the impact of energy consumption and find alternative and efficient methods to electricity. Efficient interior climate control can be used to limit the power consumption HVAC systems within apartments and homes. As an alternative to using the air conditioning system, we can harness the outside climate to help regulate the interior temperature and air quality. This will take the strain off the HVAC system, resulting in lower power consumption. A simple and common way for the outside climate to enter the feedback loop between the HVAC and room is to open a window! Opening the window a certain amount can dictate how much the exterior temperature changes the temperature inside. Using this natural diffusion of air and temperature allows for lower electricity consumption, as well as a natural cross-ventilation for the room.

Our solution is to use a window (Fig. 1) that is attached to a motor that opens and closes accordingly when given a certain desired interior temperature. The window has sensors that measure rain, temperature, humidity, and particulate matter in ppm which is transmitted to a microcontroller. The microcontroller then decides to open or close the window based on these parameters. To detect obstructions, the motor current is monitored and time it is moving.. If the current gets larger than normal, then the window will stop moving. The system is powered by a standard 120 VAC socket, and has multiple modules for the sensors, motor, microcontroller, and user interface.

Figure 1: Final product

1.2. Background

For a typical homeowner, energy expenditure becomes a costly monthly fee. If people are looking to save money, most of the time they will turn off their thermostat or match it with the outside temperature. This does help, but using heating and cooling in a building can build up cost. The typical heating and cooling can make "up about 42% of your utility bill" [1]. Electricity is just one part of the utility bill, as there are water costs and sometimes natural gas costs as well. For reference, "The average monthly residential electricity bill in Illinois is $87… and is… less than the national average of $107 per month" [2]. By providing a way to reduce the use of electricity and natural gas in temperature management, a homeowner could save hundreds of dollars a year.

For cooling savings, the best time to open a window is during the evenings and nights. If someone were to leave a window open at night, it could result in disastrous consequences if there was a nighttime storm, high humidity, or a spike in a particulate matter like pollen. The consequences could be damaged furniture, damaged paper products, high AC unit work for humidity, or allergic reactions at night and in the morning. By designing the window self-manage opening and closing itself, leaving a window open would be inconsequential as if conditions become undesirable, the window will close itself.

1.3. High-level requirements list

- Window should compare outside temperature and indoor temperature to open and close under correct conditions.
- Window should stay open/closed based on the particulate matter in the air (< 30µg/m3), humidity (< 60%), and rain.
- User should be able to adjust temperature threshold for the window, and adjust the window aperture manually if desired.
- Window should be prevented from opening or closing in the case of an obstruction.

2. Design

2.1. Block Diagram

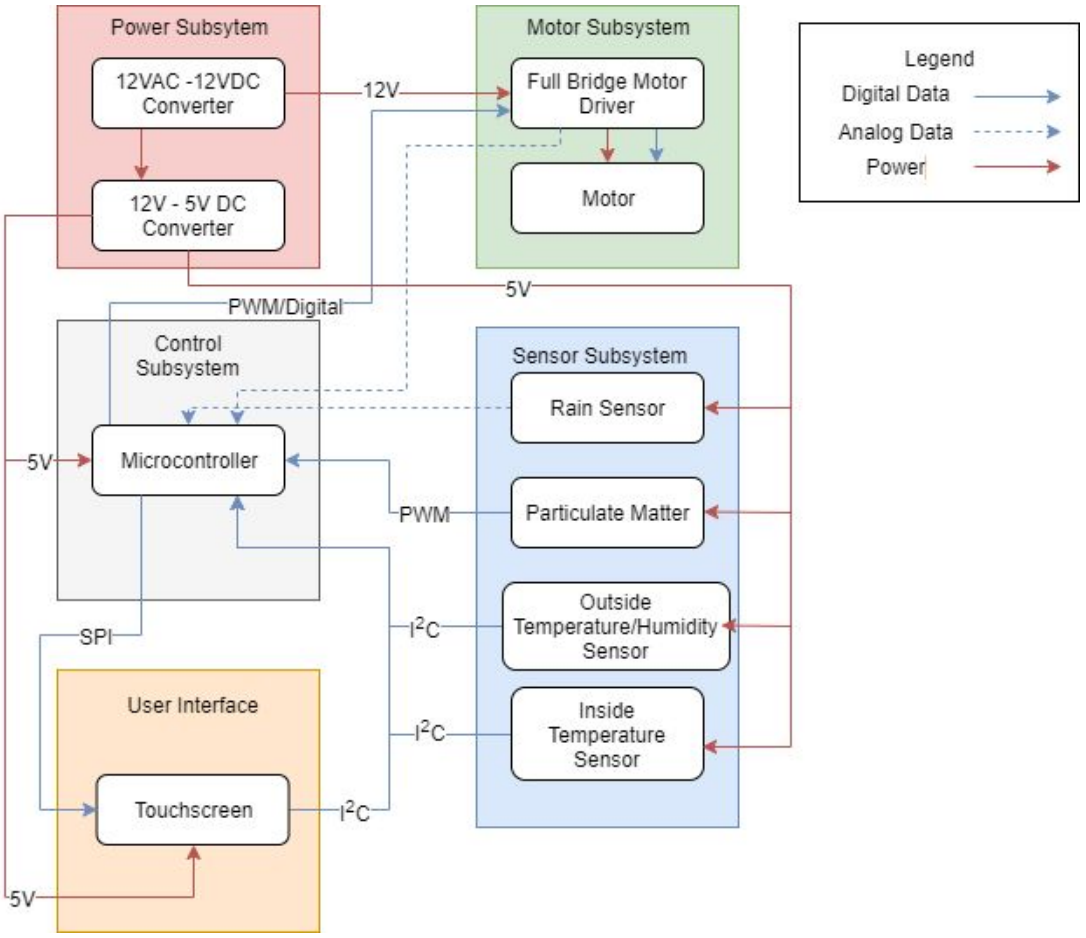


Figure 2.1: System Block Diagram

2.2. Physical Design:

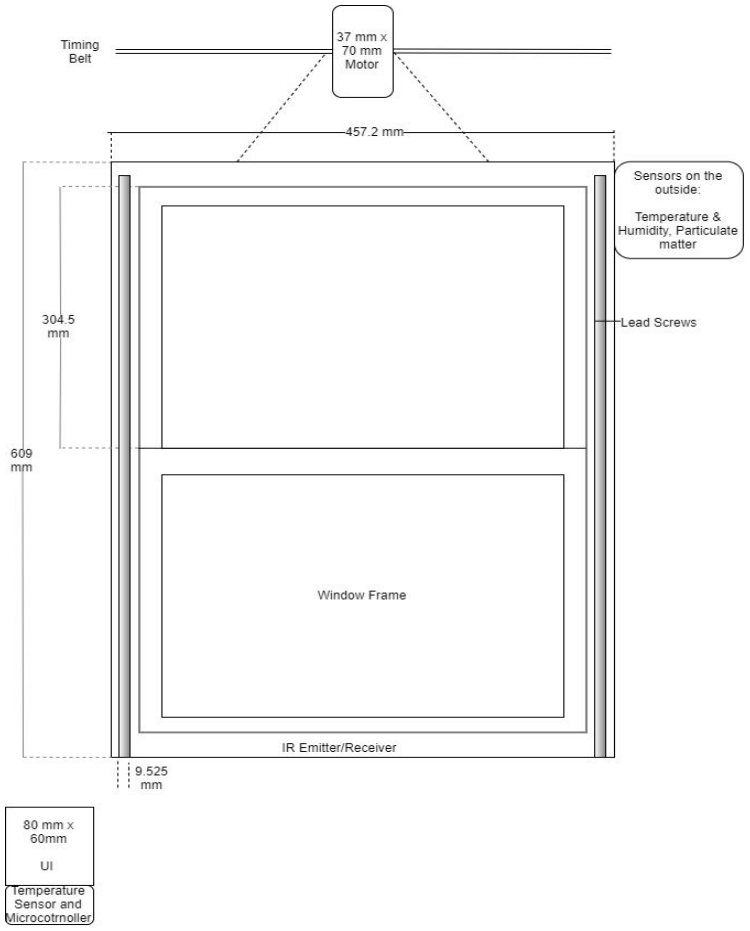


Figure 2.2: Physical Design

We have an 18 inch by 24 inch, vertically hung window. On the sides of the frame we have two lead screws that turn and carry the window frame up and down. These two screws are driven by a timing belt that is attached to a 12 V DC motor with encoder that is attached to the top of the window, on top of the left screw in Figure 2.2. On the outside of the window at the top we have an encasement that contains the temperature/humidity sensor and the particle matter sensor. Finally near the window we will have the touch screen UI along with the temperature sensor and microcontroller on the inside of the window. The sensors will be PCB mounted along with the microcontroller. The motor driver is on the PCB with the microcontroller.

2.3. Block Design

2.3.1. Power Module

The power module is required to convert 120 VAC into 12 VDC and 5 VDC to supply power to the motor driver, microcontroller, user interface, and sensors.

- 120VAC-12VDC adapter

  To supply power to the system, we used the HitLights PWR-12V-060-30-UK. It claims an efficiency of >85%, which, although not ideal, is the most efficient given the budget available.

For details regarding power and current requirements, see tolerance analysis (pg. 21)

Table 2.0: 120VAC to 12VDC Adapter Data

Input Voltage	120VAC
Output Voltage	12VDC
Output Current	8A
Output Power	96W

- 12VDC to 5VDC

  A TPS62133RGTR buck converter by Texas Instruments is used to supply a 5V rail to the control, sensor and user interface subsystems. We require a steady output voltage for this component, as most dependant components require an input voltage range of ±5%

Table 2.1: 12VDC to 5VDC Buck Converter Data

Input Voltage	3-17 V
Output Voltage	0.9 - 6 V
Output Current	3 A
Output Power	1.2 - 10 W

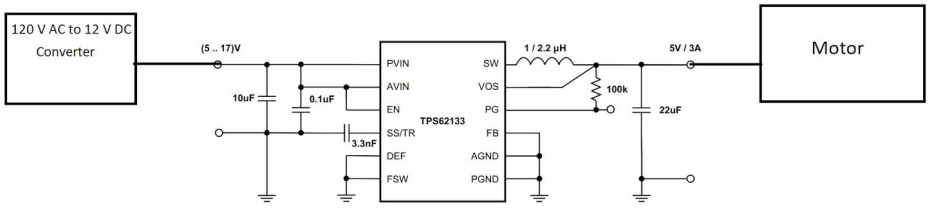


Figure 2.3 Buck Converter Schematic [11]

2.3.2. Control Module

The control module consists only of the microcontroller, which receives data from the peripheral devices, processes this information, then decides whether to activate the motor driver according to a set of predetermined parameters.

- Microcontroller

  Our design uses the ATMEGA328-PU microcontroller by Microchip Technology/Atmel. The microcontroller has 3 bi-directional ports (two 8-bit and one 7-bit), which is used to interface with the various sensors, the motor driver and the touch screen. The microcontroller reads the data provided by rain, humidity and temperature sensors and makes a decision as to whether to open or close the window (see Fig. 2.4 for details). Additionally, it uses the current sense input

(from motor driver - pins 24 & 25 in Fig) to detect obstructions, and to determine when the window is in the closed position. Data from the motor encoder will then be used to determine the relative position of the window. The IR sensors and the temperature/humidity sensor will run off a single I2C interface.

Table 2.2: Microcontroller Data

Supply Voltage	5V
Supply Current	0.2mA

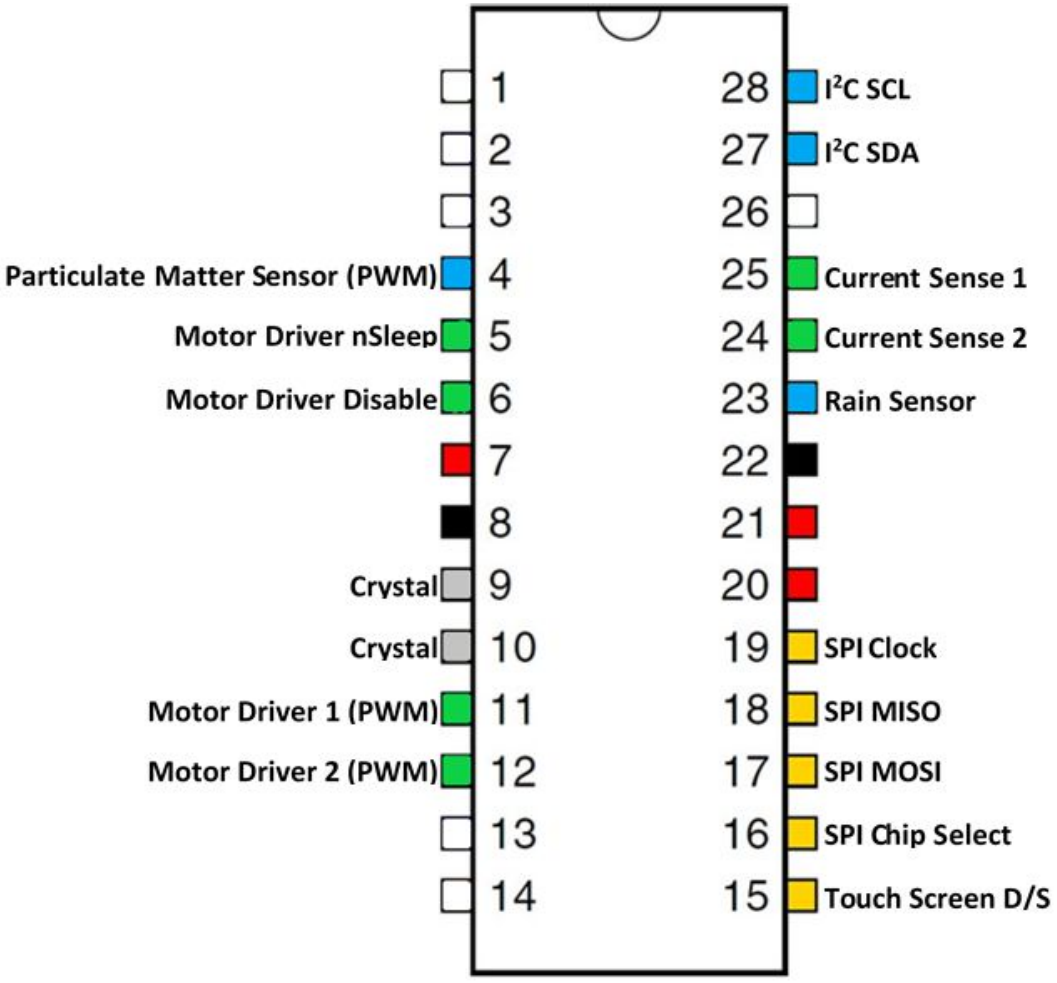


Fig 2.4: Microcontroller pinout

2.3.3. Motor Module

- Full bridge motor driver

  To provide power to the motor, we used the DRV8873SPWPR full bridge motor driver by Texas Instruments. It provides ±12V to the motor, depending on the input from the microcontroller, at up to 10A, which is sufficient to handle the peak motor current draw of 5.5A. Additionally, it has a current sense data output, which goes into the microcontroller as a detection for obstructions. If there is something

blocking the window, load on the motor increases, so there will be a larger current draw from the motor driver.

Table 2.3: Motor Driver Data

Supply Voltage	4.5 - 38 V
Logic Inputs	5 V
Output Current	10 A

- 12V Brushed DC Motor

  We used the motor with part number 4754 from Polulu. This motor is a 12 V brushed DC geared motor with a gear ratio of 70:1. The force required to pull the window up is equivalent to lifting a mass of 1 kg. The gear that is attached to the motor to go on the timing belt has a diameter of 5.5 cm. The calculation for the minimum amount of torque is:

  Torque from the screws:

  $N \times \mu \times (Clamp\ force)$

  $(1 \times 9.8) \times 0.2 \times 9.525 \ = \ 18.669\ N * mm$

  Convert to Kg:

  $18.669 \times 0.102 = 1.9043\ Kg * mm$

  For maximum efficiency, the motor can operate at a torque at or below 32 Kg*mm, and the torque needed is much less at a window weight of 1 Kg. The friction of the window is not accounted for, but there is a very small amount of friction pulling the window up, so it is negligible. At maximum efficiency, the motor runs at 130 RPM, which is an appropriate speed, as the screws move ⅛ of an inch every rotation, which is 41.275 cm every minute. Since there is not much movement once the window is set, it does not need to move large distances instantaneously, so this speed is sufficient.

Table 2.4: Motor Data

Gear Ratio	70:1
No Load Speed	150 RPM
No Load Current	0.15 A
Stall Current	5.5 A
Maximum Power	10 W
Stall Torque	270 Kg*mm
Torque @ Max Efficiency	32 Kg*mm
Speed @ Max Efficiency	130 RPM

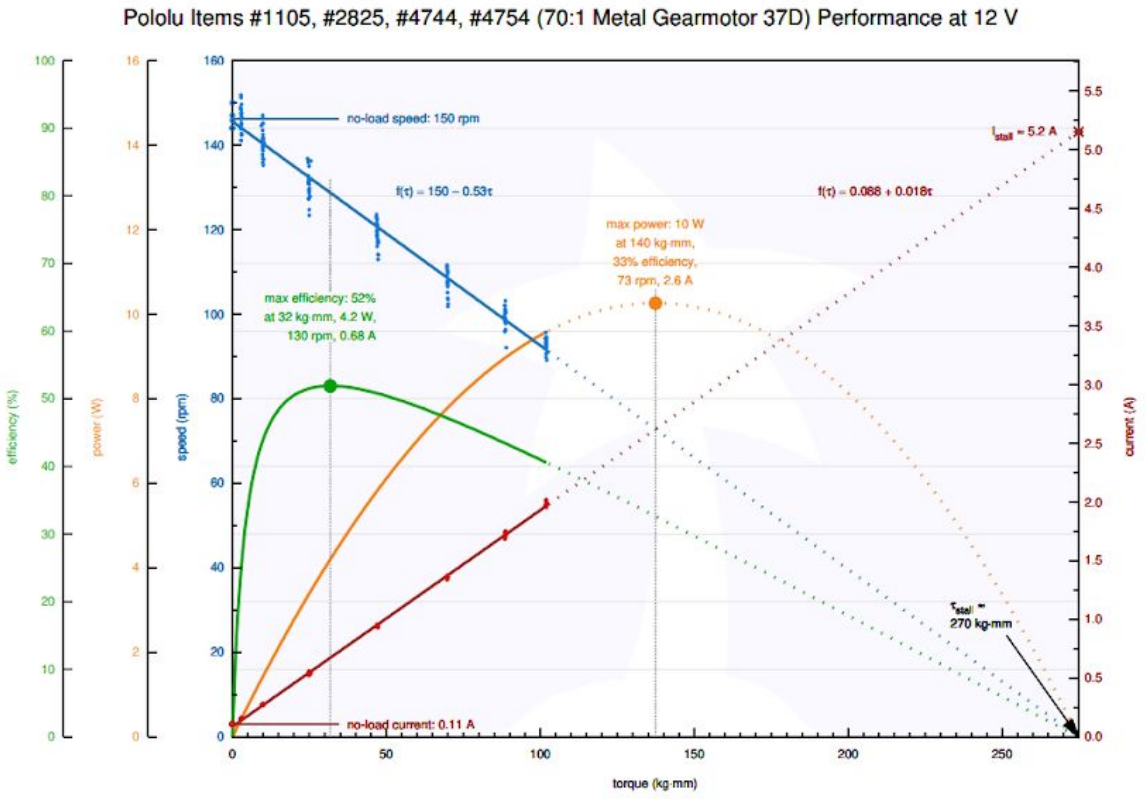


Figure 2.7: Performance characteristics of motor [9]

2.3.4. User Interface Module

The user interface allows the operator to either control the window position manually or set it to automatic mode, in which case the window will open and close according to the measurements made by the sensors.

- Touch Screen

  The original idea for the user interface consisted of a simple LCD screen with a few physical buttons, however, for a similar cost, we were able to implement an LCD touch screen instead. The touch screen we used is the Adafruit 2.8" and 3.2" Color TFT Touchscreen Breakout v2**.** The touchscreen receives data from the microcontroller using SPI and sends data via I2C. The screen is powered by 5 V and its current draw is 220 mA.

  The layout of the UI is displayed in Figure 2.5. The outside and inside temperatures are displayed on the top with the desired temperature in a larger font. In the middle is the toggle for auto and manual mode. In manual mode, one can use the arrows on the left to move the window up and down. In auto mode, one can use the set temperature arrows to the right, which raises/lowers the desired temperature. The touch screen also displays a warning in the case of rain, bad air quality, and/or an obstruction in the path of the window.

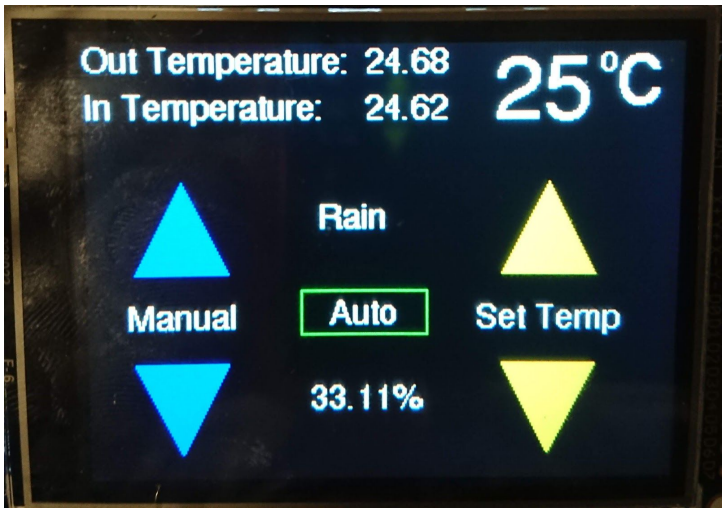


Figure 2.5 - User Interface Layout

2.3.5. Sensor Module

The sensor module has all the sensors for our system and relays information about the environment back to the microcontroller. The information comes from an indoor temperature sensor, an outdoor temperature and humidity sensor, particulate matter sensor, and rain sensor. This data is then used by the microcontroller to make a decision as to whether the window should be open or closed

- Temperature Sensor
  We have an indoor temperature sensor, which is model mcp9808 from Adafruit. We originally had a temperature sensor that would give us a varying voltage analog signal that corresponds with temperature, but it ended up giving us very high values and was unusable. We switched to a temperature sensor that used I2C instead. Although there were a lot of different registers in this device such as temperature thresholds, we only used the ambient temperature register at address 0b.

Table 2.6: Temperature sensor data

Input Voltage	2.7-5.5 V
Current	200 μA
Temperature Range	-40-125 C (-40-257 F)
Accuracy	+/-0.5 C (0.9 F)
Output	I2C

- Temperature and Humidity Sensor
  For the outdoor temperature and humidity sensor, we used HDC1080DMBR by Texas Instruments. Since indoor temperatures do not need a humidity check since AC keeps humidity at a certain level, it is not needed. We do however, need to check the outside humidity. Checking

humidity is important since humidity can do a lot of damage on property as well as health depending on the level. High humidity will also increase the AC run time, which would increase electricity usage. This is a good temperature sensor because it is very accurate, only deviating by +/- 0.5 C. The humidity sensor is very accurate as well, deviating only by +/- 2 % Relative Humidity. Although the humidity sensor does have an age to it, it only changes by +/-0.25 % RH per year, which is a very slow decline.

This temperature and humidity sensor is good because it has a large operating temperature for both the temperature sensor and the humidity sensor, having a range of -20-85 C (-4-185 F). This range is fine for operating an automatic window, but when it is off, the range increases to -65-150 C (-85-302 F). This chip can also run at 5V, but has a large voltage range of 2.7 to 5.5 V. The output of the chip has an I2C serial data line. Although there are many registers, we only used the first two registers at 0x00 and 0x01 since those are the data registers for temperature and humidity. The rest of the registers are used for ID and status. This chip is connected to our own PCB, acting as a breakout board.

Table 2.7: Temperature/Humidity sensor Data

Input Voltage	5 V
Current	Sleep Mode: Average 150 nA 1 μA at highest temperature Active Mode: 125-250 μA depending on temperature and Vdd
Temperature Range	-20-70 C (-4-158 F)
Accuracy (Humidity)	+/-2 %RH
Accuracy (Temperature)	When 5 C < T < 60 C: +/-0.1 C F: When 41 F < T < 140 F: +/- 0.18 F
I2C Clock Frequency	10- 400 kHz
I2C Clock Low Time	≥ 1.3 μs
I2C Clock High Time	≥ 0.6 μs

- Rain Detector

The rain detector is a model a13082300ux1431 from Uxcell. Most rain detectors have similar form of some kind of flex resistor. We chose this rain detector because the price is low and the sensitivity can be adjusted using a potentiometer. The rain detector is necessary because we

need to check to see if it is raining outside before a window can be opened. When rain starts, humidity will not increase until a while later, so a rain detector is a must with a humidity sensor.

This rain detector also comes with its own board, which can output a digital switch output or an analog output. We used the analog output of the board. The sensor sends out a high signal when no rain is detected and a low signal when there is rain. Since the maximum recommended voltage setting is 5 V, we run all the sensors at 5 V. Since the Arduino Uno is a ten bit resolution for analog to digital conversion (0-1023), we designate a value of less than 950 to be when there is rain.

Table 2.8: Rain sensor data

Input Voltage	3.3-5 V
Input Current	$\leq$15 mA
Output	Active Low Digital Output

- Particulate Matter Sensor

The particulate matter sensor is a dust sensor (Model PPD42NS by Shinyei Corporation). The particulate matter sensor is on the outside of the window. The function of this sensor is to detect particulate matter levels outside like dust and pollen, things that are detrimental to the inside environment. This sensor was chosen because it is robust since it can handle temperatures from 0-45 C (32-113 F). Although the values are not very large, for extreme temperature values, this device is not expected to be on. When off, the device can be in a range from -30-60 C (-22-140 F), which is a range that fits most climates. It also can detect particles larger than 1 μm, which is a good range for detecting pollen and dust (e.g. the size of pollen is 6 μm).

The device operates in voltages between 4.5-5 V. The output of this device goes straight into the microcontroller since it is a digital signal (PWM). Based on the concentration of particles, the duty cycles hits a low pulse occupancy percent up to 16%.

Table 2.9: Particulate matter sensor data

Voltage Input	4.5-5.5 V DC
Current	Up to 90 mA
Operating Temperature	0-45 C (32-115 F)
Output	PWM wave, negative logic High: > 4 V Low: < 0.7 V Unit Wave time: 30 sec

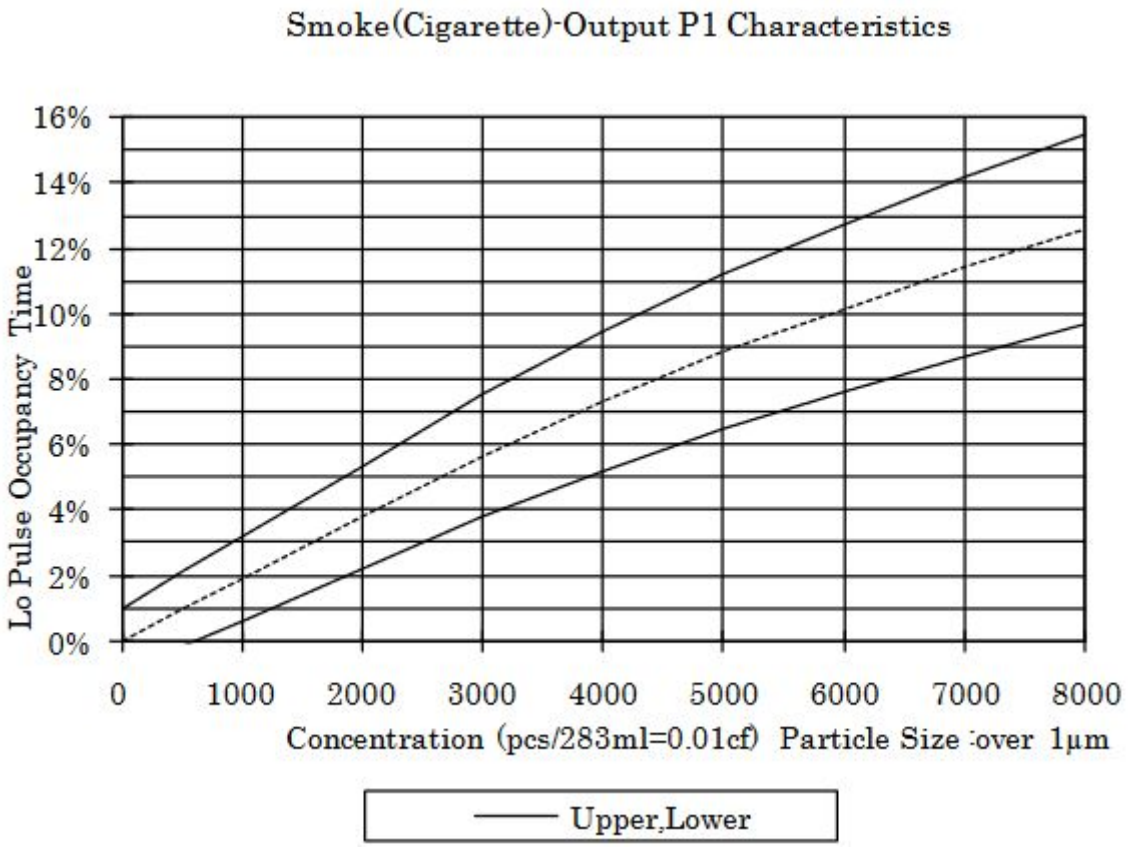


FIgure 2.6: Particulate matter sensor characteristics [4]

2.4. Requirements and Verification

A detailed requirements and verification table can be found in appendix A

2.4.1. Power Module

We tested two parts of the power module. The first one was the 120 V to 12 V converter. When we plugged the converter into the wall and attached it to the PCB, the power line connected to the converter was probed using a multimeter. The multimeter displayed a voltage of 11.983 V, very close to 12 V and within specifications and requirements of this part.

The second part that we tested was the 12 V to 5 V step down converter. The first time we tried testing the chip, we had the same voltage at the output and input of the chip. We thought that the chip might have been blown because of the high power that the 120 V to 12 V converter can provide. The current delivered by the 120 V to 12 V converter has a maximum of 9 A while the 12 V to 5 V converter can only take a maximum of 3 A, so if there was a short, it would have blown the chip. We bought a second chip and although we did not get the same error as the first chip, we got an output of 1-2 V instead of the 5 V that was needed. The output, input, and ground were not shorted together and the soldering job was fine. The issue would have been the chip, but we ran out of time to buy more and experiment.

2.4.2. Control Module

For the microcontroller, we checked if the signals from the sensors were being read. For the two sensors that required I2C, we used test code that just

called the address of the I2C device as well as read/write values to some of the registers. This worked, so our I2C line worked fine. With our analog rain sensor, we used analogread() on the Arduino to read the voltage coming from the board and cross checked it with a multimeter. The two readings were correlated and we ruled the analog signal to be working and was being read. The digital signal from the particulate matter sensor was checked by outputting what was being sent in the digital pin of the arduino and cross checking the output of the sensor with an oscilloscope. When there was a high recognized by the microcontroller, there was also a high on the oscilloscope, so the digital input to the microcontroller worked.

Since we know the I2C line worked, we can safely assume that any errors coming from any output of the touchscreen would not be an error with the microcontroller, but with the hardware setup of the touchscreen itself. The SPI interface test for the touchscreen (the display) was tested by the procedure mentioned in the user interface module below.

2.4.3. Motor Module

The two things that we tested is the motor and the motor driver. For the motor, we were able to hook up the power to 12 V. We tested the operation of the motor by putting 12 V in each input. We timed the window opening and closing, and the window was moving at a consistent time. The window completed a full open/close in 25 seconds, which is completely reasonable. This meets our requirement of opening/closing the window within 30 seconds.

There were three outputs that we needed to check for the motor driver. We needed the voltage to be correct and within specifications and we needed the current to be outputted correctly and within specifications as well. By probing with the motor driver output with a digital multimeter, we found the voltage coming out to be 11.856 ±0.05 V. This is completely within our requirements of ±5% of 12 V. The motor driver has two outputs that output the current the motor driver is drawing, so we attached these two to analog inputs to the microcontroller and printed out the current using test code. When we ran the motor driver with the motor, the current draw was at an average of 0.53 ±0.03 A. When tested on different resistor loads of resistances up to 80 Ω, the current never went above 0.2 A. Given that the current should not go above 5.5 A, we also met our requirements for voltage and current.

2.4.4. User Interface Module

For our UI interface, there were two parts that we tested. We tested the SPI portion using test code that would change the screen to black instead of the bright white that appears when on. The screen successfully turned completely black, so the SPI interfacing worked with the screen.

For the touch portion, we had test code with I2C interfacing that would print out the location of the touch if one existed. The code was able to display locations that changed logically depending on where our finger was. There was no response when we touched the bezel of the touch screen and every point on the touch screen had a response. With both SPI and I2C working, we labelled the UI Interface as working.

2.4.5. Sensor Module

The first sensor tested was the rain sensor. We had the rain sensor connected to 5 V from a power supply and the analog output to a digital multimeter. When there was no water on top, there was 4.73 V. When water was added, voltage dropped to 0.46 V. We measured the water needed to switch the signal low, and we needed 1 mL of water before the rain sensor recognized rain. There was a potentiometer to adjust sensitivity, but we believed 1 mL was an acceptable amount of rain to detect before the window closes, since a raindrop is roughly 0.05 mL, so if twenty drops of water fell on the board, it would detect rain. We believe that is enough to detect light rain.

Following this, we tested the temperature sensors. We brought in a digital thermometer and tested both temperature sensors using our microcontroller to display the values from I2C. The temperature sensors differed from the thermometer by a maximum of 0.37 C, which is well in the specifications.

The particulate matter sensor did not have an exact test, as it is very hard to classify particulate density in the current air without using another sensor. We tested this matter sensor with sample code that would output the density of particulate matter in thirty second cycles. We dusted the air with blackboard erasers for thirty seconds and noticed an increase of particulate density. At the beginning, the density never rose above 1000 ppm, but with the chalk dust, the sensor consistently hit over 17000 ppm.

We had two testing verifications that were completely off specifications. One of them was the IR sensors that we had originally planned to use in our design document. The IR sensors were supposed to give a value that differed based on how far away an obstruction was. By using test code to output the distance from the I2C output, we only got random values. The values were varied and did not change variance when we tried to put an obstruction in front of the IR sensors. We tested three IR sensors with no luck, and wrote them off as completely broken.

The other sensor that ended up completely off the requirements was our original temperature sensor. It was supposed to be voltage analog signal that differed based off of temperature, but when we connected the voltage, we got a voltage of 1.5-2 V, which corresponds to around a temperature range of 55-80 C (131-176 F) (Fig. 7). Since we measured this at room temperature, this data proved the sensor did not work and the sensor was replaced.

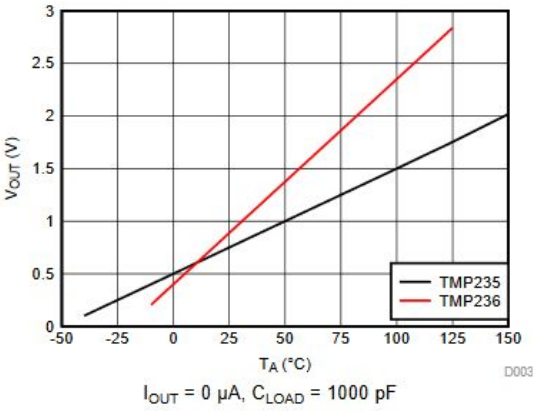


Figure 2.7: Original Temperature Sensor Voltage Data

2.5. Control Description

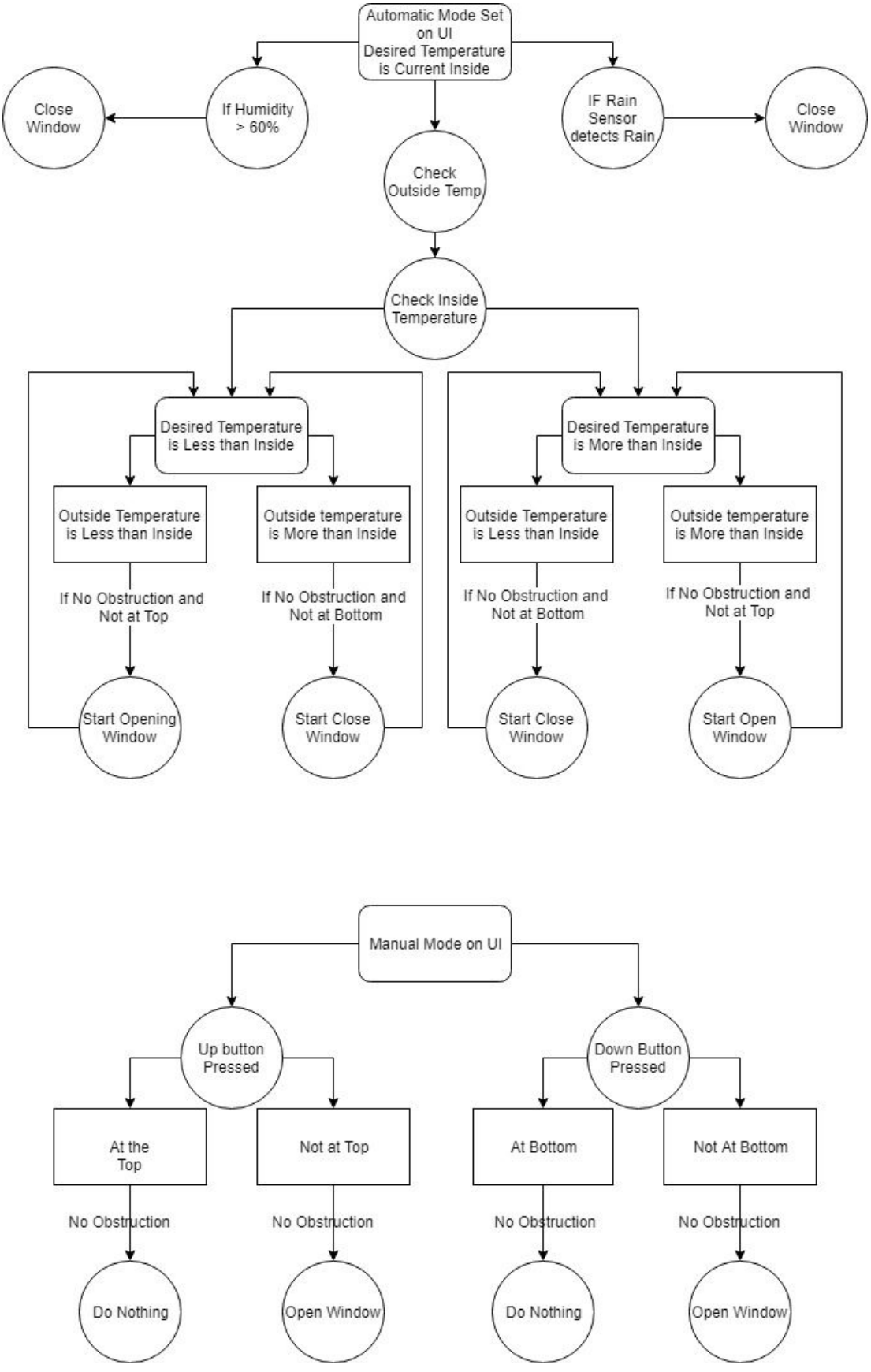


Figure 2.8: System operational flow diagram

In our design (Figure 2.8), there are two different modes that can be processed by the microcontroller. For the first mode (automatic), we set a desired temperature on the UI, then get data from both the inside and outside temperature sensors. Depending on if the desired temperature is more or less than what we desire, we either close or open the window. If there is precipitation outside then the entire system goes and keeps the window closed to make sure that the water is not entering the facility. If there is no precipitation, the control module continues on. If it is colder outside than it is inside and the temperature we want is less than the current inside temperature we will open the window, if it is warmer outside than inside then we will close the window. On the other hand if the desired temperature set is more than the current if it is warmer inside than outside we will close the window and if it is warmer outside than it is inside, we will open the window. Through this entire process we will keep track of the position of the window. This way we can detect if the window is at the maximum or minimum position and we can stop the motor from moving the frame. Finally, another check we constantly will be doing is that if if the IR detects and obstruction on the window, then we will halt all processes and stop the motor from running and moving any direction. We plan to send data to the UI to show that there is obstruction, stopping the window from moving up or down.

2.6. Tolerance Analysis

The power module of this system is critical to its success, as all other modules rely on it to function. Additionally, the main purpose of the system is to save energy, and the largest source of power loss will be in the 120VAC to 12VDC adapter, so it is essential that this component satisfies efficiency requirements.
The 120VAC to 12VDC adapter is rated to 8A. If the current draw exceeds this rating, it would be detrimental to the system, causing potential overheating, which may be a fire hazard.

Table 2.10: System current draw breakdown

Subsystem	Current Required (mA)
Sensors	0.03
User Interface	220
Motor	530
Microcontroller	2
Total:	752

As seen in Table 2.10, the maximum total system current draw is 752 mA, which is 7.248 below the power adapters' maximum rating of 8A. This gives us a margin for error of 93%. Table 2.10 contains a breakdown of the power consumption of all major system components.

Table 2.11: System power consumption breakdown

Component	Power Consumption (W)
120 VAC to 12 VDC adapter	0.02*
Sensors	0.15e-3
User Interface	1.1
Motor	8e-3**
Microcontroller	0.01
Total	1.2

Assuming motor is activated on average 4 times a day for 30 seconds (to fully open or close), drawing an average of 0.500A (up) and 0.560A (down) the power consumption of the adapter is given by: (1 - efficiency)($VI_{active}T_{active}$ + $VI_{idle}T_{idle}$)/24 hours

**Assuming motor is activated on average 4 times a day for 30 seconds (to fully open or close), the power consumption is given by: $VI_{active}T_{active}$/24 hours

A typical HVAC system functioning in fan only mode will use up to 500 W [10], and runs for an average of 9 hours a day [11]. The total daily energy usage, based on these numbers, comes to 4.5 kWh. In comparison, the window system uses a total of 0.0288 kWh a day, resulting in a reduction of 4.47 kWh. The HVAC system will still use its cooling and heating functions, as this is beyond the capabilities of the window system, so the heating and cooling will still have a sizeable contribution to total household energy consumption. However, based on an average monthly electricity consumption of 767 kWh [11], the use of the automated window system results in a reduction of 17% of a household electricity bill, a saving of around $15 a month.

3. Cost and Schedule

3.1. Cost Analysis

3.1.1. Labor

Based on an average graduate salary of $76,079 [7] for UIUC electrical engineering graduates, our hourly rate is estimated to be $35/hour per person. Assuming 10 hours of labor per person per week, and 16 weeks of development, the total labor cost will be:

$$\$35 * 10\ hours/week\ *\ 10\ weeks\ *\ 2.5\ *\ 3\ members\ = \$26\ 250$$

3.1.2. Materials

Table 3.1: Material cost breakdown

Description	Manufacturer	Part No.	Quantity	Cost
Sensor Subsystem				
Board Mount Temperature Sensor	Adafruit	Adafruit MCP9808	1	$4.95
Board Mount Humidity and Temperature Sensor	Texas Instruments	HDC1080DMBR	1	$2.80
Rain Sensor	Uxcell	a13082300ux1431	1	$14.16
Particulate Matter Sensor	Seeed Studio	101020012	1	$18.50
Motor Subsystem				
DC Geared Motor with 64-bit Encoder	Polulu	4754	1	$39.95
Full Bridge Motor Driver	Texas Instruments	DRV8873SPWPR	1	$4.04
Control Subsystem				
Microcontroller	Microchip Technology	ATMEGA328-PU	1	$1.95
Power Subsystem				
120VAC - 12VDC Converter	HitLights	PWR-12V-060-30-UK	1	$26.69
12V - 5V DC Buck Converter	Texas Instruments	TPS62133	1	$2.15
User Interface				
LCD Touch Screen	Adafruit	N010-0554-T703	1	$29.99
Other				
Window*	-	-	1	$80.00
Mechanical Parts*	-	-	-	$20.00
Miscellaneous Electrical Parts	-	-	-	$10.00
PCBs	-	-	3	$30.00
TOTAL				$267.71

* Window and mechanical part costs are covered by the machine shop, and thus are estimated to give a more accurate material cost.

3.1.3. Grand Total

Taking into account both the labor and material costs, the grand total for this project will be $26517.71

3.2. Schedule

Table 3.2: Project Schedule

Week	Derik	Hersh	Louis
9/16	Research Sensors	Research Motors	Research Power
	Research component specifications	Research component Specifications	Research window and motor specifications
9/23	Research component specifications	Research component specifications	Research window and motor specifications
	Start design documentation	Start design documentation	Start design documentation
9/30	Finish design documentation	Finish design documentation	Finish design documentation
	Place Parts Order	Find exact parts to order	Find exact parts to order
10/7	Research PCB TTL chips	Order all leftover parts	Test any parts in this week
10/14	Start PCB design	Start PCB design	Start PCB design
	If applicable, start PCB assembly If not, Test new components	Test motor subsystem on window	Test all of the new components
10/21	Finish motor system on window	Finish PCB assembly	Start adding sensors to the window
10/28	Add user layout to the window	Program the microcontroller	Finish adding all the sensors
11/4	Test user layout on window and systems	Program the user layout	Start debug of sensors and motor subsystem
11/18	Debug programs, sensors, and motors	Debug programs, sensors, and motors	Debug programs, sensors, and motors
11/25	Thanksgiving Break	Thanksgiving Break	Thanksgiving Break
12/2	Final Demo Preparation	Final Demo Preparation	Final Demo Preparation
12/9	Final Report	Final Report	Final Report

4. Conclusion

4.1. Discussion of Ethics and Safety

We ensured that our project will take into account the safety and concerns of the user. The ethical side of our project will be to maintain the veracity of our claims that this project will help save energy. Because this is an automated system which runs unsupervised, we are transparent with safety issues and warn the user to follow the IEEE code of ethics; more specifically, the first point where we "disclosed promptly factors that might endanger the public or the environment," [5]. Another IEEE Code of Ethics point that we will follow is "avoid injuring others, their property, reputation, or employment by false or malicious action," [5]. We ensured that the window had sensors and overrides any action to stop movement in case of obstruction.

Another safety concern we had is the sensors we have are electronic and may be exposed to hazardous weather. Water can cause the circuitry to short circuit, which presents a fire hazard and risk of electrocution, which could cause damage to the home, window and anyone close by. We made sure that all circuits and sensors are properly shielded from any potential situations that could lead to this concern.

Another ethics concern we saw whether our promise of reducing energy consumption in homes is true. We want "to be honest and realistic in stating claims or estimates based on the available data;" [5]. Our main goal was to make an environmentally friendly window system to reduce power consumption homes and advertise the product as such only if it truly does curb wasted energy. Considering that the typical HVAC unit uses up to 5000W [10], and this system uses an average of 0.12W (see tolerance analysis for detail, pg. 22),

4.2 Future Work

For future work we wanted to encase all the sensors and microcontroller in a container so that it couldn't be damaged by hazardous weather. We would also make sure that the motor is not as noisy to reduce noise pollution in homes while the window was moving up and down. Our main goal of this project was to implement it into an HVAC system or smart home system. What this allows us to do is connect to the fan and heater in the HVAC and make sure that it shuts off in accordance to our control flow. THis would allow the home to save even more energy and money. Finally, we would like to make sure that our microcontroller gets of the arduino and on another PCB, as this was one of the requirements we could not fulfill. Instead we had to opt to keep the microcontroller on the arduino development board.

4.3 Accomplishments

We learned a lot about how to work as a team and brainstorm an idea and make it come to fruition, We were able to get a window that safely opened and closed and also responded to readings from all our sensors. There were some troubles as we couldn't get the IR sensor or motor encoder to work. Thankfully due to some quick thinking we were able to use the current draw to make the window safe and secure. We were unable to get out microcontroller PCB working nor the 12 V to 5 V converter and had to resort to using the entire Arduino UNO and it's 5 V pin to make sure our window worked. We learned a lot about different forms of communication and electrical engineering principles.

References

[1] Energy.gov. (2019). *Home Heating Systems*. [online] Available at: https://www.energy.gov/energysaver/heat-and-cool/home-heating-systems [Accessed 12 Sep. 2019].

[2] Electricity Local. (2019). *Champaign, IL Electricity Rates*. [online] Available at: https://www.electricitylocal.com/states/illinois/champaign/ [Accessed 12 Sep. 2019].

[3] Texas Instruments. (2019). *TMP23x Low-Power, High-Accuracy Analog Output Temperature Sensors.* [online] Available at: http://www.ti.com/lit/ds/symlink/tmp235.pdf [Accessed 01 Oct. 2019]

[4] Seeed Studio. (2019) *Grove - Dust Sensor.* [online] Available at: https://www.seeedstudio.com/Grove-Dust-Sensor-PPD42NS.html [Accessed 01 Oct. 2019]

[5] IEEE.org. (2019). *IEEE Code of Ethics*. [online] Available at: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 13 Sep. 2019].

[6] US EPA. (2019). *Particulate Matter (PM) Basics | US EPA*. [online] Available at: https://www.epa.gov/pm-pollution/particulate-matter-pm-basics [Accessed 16 Sep. 2019].

[7] Illini Success, (2018), *Illini Success Annual Report 2017-2018*, pg. 19. [online]. Available at: https://uofi.app.box.com/s/ml9oh48vawrhm7e019kw5ix4j91j605p [Accessed 28 Sep. 2019]

[8] Elegoo.com. (2019) *ELEGOO UNO R3 2.8 Inches TFT Touch Screen*. [online] Available at:https://www.fujitsu.com/downloads/MICRO/fcai/touchpanels/control-board-single-input.pdf [Accessed 01 Oct. 2019]

[9] Pololu Robotic & Electronics. (2019). *37D Metal Gearmotors,* [online] Available at: https://www.pololu.com/file/0J1706/pololu-37d-metal-gearmotors.pdf [Accessed 10 Oct. 2019]

[10] USInspect. (2019. *Can Running the HVAC Fan Continuously Save Energy?*. [online] Available at: https://www.usinspect.com/blog/can-running-hvac-fan-continuously-save-energy-costs-part-1-3/ [Accessed 9 Dec. 2019]

[11] EnergyUseCalculator.com. (2019). *Electricity Usage of a Central Air Conditioner,* [online] Available at: http://energyusecalculator.com/electricity_centralac.htm [Accessed 16 Sep. 2019]

[12] TI.com. *(2019) TPS6213x 3-V to17-V, 3-A Step-Down Converter In 3x3 QFN Package*, [online] Available at: https://www.ti.com/lit/ds/symlink/tps62133.pdf [Accessed 3 Oct. 2019]

Appendix A

Module	Requirements	Verification
120V-12V Converter	Convert 120VAC to 12VDC from a wall plug with ≥85% efficiency	1. Connect adapter to wall outlet 2. Measure input and output voltage and current using DMM 3. Calculate efficiency
12V-5V Converter	Converts 12VDC to 5VDC with ≥95% efficiency	1. Connect converter to 12VDC supply 2. Measure input and output voltage and current using DMM 3. Calculate efficiency
	Maintains steady output under all load conditions (within 5% 5V at output)	1. Attach variable load to output 2. Observe output behaviour on oscilloscope with varying load, including during startup and shut down (simulated by switching power source on and off)
Microcontroller	Responds correctly to all given input combinations ● Temperature ● Humidity ● Rain ● Obstructions (IR sensors) ● Current window position	1. Check logic using computer simulations on code 2. Connect input pins to relevant simulated interfaces (I2C, Analogue, PWM) 3. Connect oscilloscope to motor driver output pin 4. Vary inputs according to Table A.1 and measure output using DMM 5. Check that output matches control flowchart
	Interfaces correctly with touch screen	1. Connect touch screen to microcontroller via I2C interface 2. Check logical response at output of microcontroller for any functional touch inputs

Temperature Sensor (indoor)	Measures temperature to within ±1 °C	1. Place sensor in small air conditioned room 2. Allow AC to reach thermostat set temperature 3. Measure output of temperature sensor using oscilloscope 4. Vary AC temperature to extremes 5. Check sensor matches thermostat temperature
Temperature/Humidity Sensor (outdoor)	Measures temperature to within ±1 °C	1. Place sensor in small air conditioned room 2. Allow AC to reach thermostat set temperature 3. Measure output of temperature sensor using oscilloscope 4. Vary AC temperature to extremes 5. Check sensor matches thermostat temperature
	Measures humidity to within ±5%	1. Place sensor in various environments of different humidity levels (e.g. outside, room with HVAC, sauna) 2. Measure actual humidity using sensor of known accuracy 3. Read data from I2C interface and compare to measured value
Rain Sensor	Detects light rain with accuracy ≥80%	1. Spray sensor with spray bottle on mist setting 2. Check output with DMM
	Detects heavy rain with accuracy ≥95%	1. Drop single drop of water onto sensor 2. Check output with DMM
Particulate Matter Sensor	Measures particulate matter content to within ±5µg/m3	1. Place sensor in various environments of different known PM levels 2. Measure actual PM concentration using sensor of known accuracy 3. Read data from I2C interface and compare to measured value

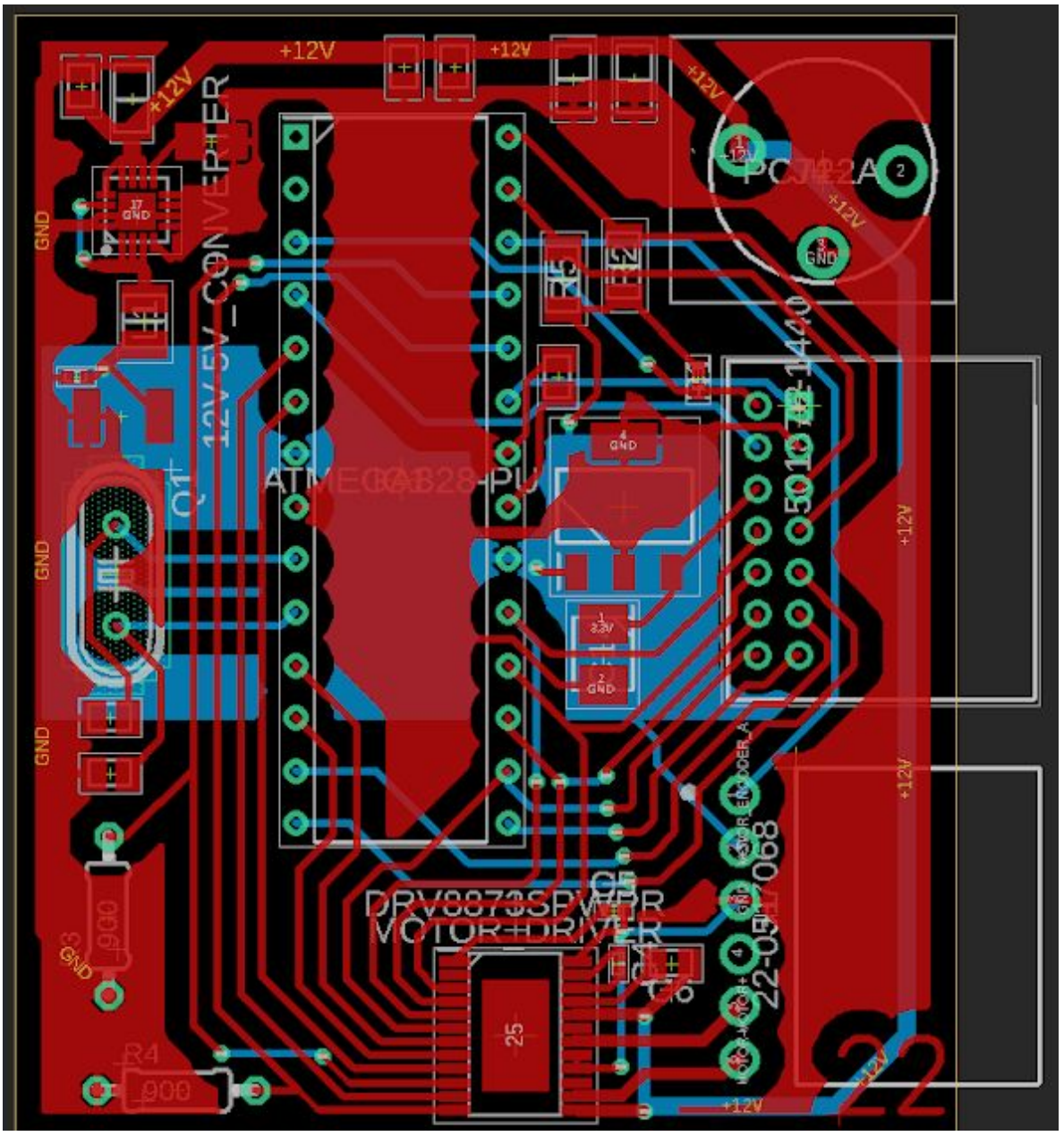
Motor Driver	Outputs ±12V (within ±5%) current up to 5.5A (stall current of motor)	1. Attach driver to 12VDC source 2. Attach variable load to output 3. Apply logic inputs for forward, reverse and brake functions 4. Vary load between 2.2Ω and 80Ω to simulate different motor operating modes 5. Use oscilloscope to observe output
Motor	Opens and closes window 25cm in 30 seconds	1. Attach motor to power supply and window set up 2. Measure time taken for window to fully open from closed position 3. Measure time taken for window to return to closed position
Touch Screen	Displays information described in Figure 2.6	1. Configure display with power source and microcontroller 2. Check spacing and color of display
	Responds to touches accurately	1. Configure display with power source and microcontroller 2. Touch all buttons and check output to see if it matches 3. Touch all non functional parts of the screen, ensure no response

Table A.1 - Microcontroller testing

Inputs			**Outputs**		
Pin	Description	Value	Pin	Description	Value
	Inside Temperature Outside Temperature Desired Temperature Humidity Rain Motor Current 1 Motor Current 2	25 °C 28 °C 25 °C 0% No 0 A 0 A		Motor nSleep: Motor Enable: Motor PWM 1: Motor PWM 2:	LOW HIGH LOW HIGH

	Inside Temperature Outside Temperature Desired Temperature Humidity Rain Motor Current 1 Motor Current 2	28 °C 25 °C 25 °C 0% No 0 A 0 A		Motor nSleep: Motor Enable: Motor PWM 1: Motor PWM 2:	LOW HIGH HIGH LOW
	Inside Temperature Outside Temperature Desired Temperature Humidity Rain Motor Current 1 Motor Current 2	25 °C 25 °C 25 °C 70% No 0 A 0 A		Motor nSleep: Motor Enable: Motor PWM 1: Motor PWM 2:	LOW HIGH LOW HIGH
	Inside Temperature Outside Temperature Desired Temperature Humidity Rain Motor Current 1 Motor Current 2	25 °C 25 °C 25 °C 0% Yes 0 A 0 A		Motor nSleep: Motor Enable: Motor PWM 1: Motor PWM 2:	LOW HIGH LOW HIGH
	Inside Temperature Outside Temperature Desired Temperature Humidity Rain Motor Current 1 Motor Current 2 NB: Motor opening	25 °C 25 °C 25 °C 0% No 500 mA 0 A		Motor nSleep: Motor Enable: Motor PWM 1: Motor PWM 2:	HIGH LOW LOW LOW
	Inside Temperature Outside Temperature Desired Temperature Humidity Rain Motor Current 1 Motor Current 2 NB: Motor closing	25 °C 25 °C 25 °C 0% No 0 A 500 mA		Motor nSleep: Motor Enable: Motor PWM 1: Motor PWM 2:	HIGH LOW LOW LOW

Appendix B


+12V
GND
ATMEGA328-PU
DRV8873SPWPR
MOTOR DRIVER
12V 5V CONVERTER