Active Windows

Jonathan Clifford, EE, Frank Cremonini, CSE, Griffin Manns, EE,
& Connor Moore CSE

Abstract - As climate change becomes ever more apparent in the daily lives of people, every effort must be made to combat it in any way possible. The average home allows for a great deal of potential automation, especially with regards to windows, where allowing for a "smart" or "active" window ensures that there is minimal upkeep in terms of energy consumption, while still making it easy for homeowners to control their windows. In our proposed solution, this group focused on allowing for remote control of a window using an Android Phone Application, and developed a model test window which allows for actuation regardless of whether the user is located within their home, or elsewhere. This first step of home automation ensures that future engineers, contracted by the Manhattan2 Initiative will be able to build upon the system that we designed.

I. Introduction

With climate change becoming ever more relevant, engineers from every discipline are scrambling to combat the issue. Through methods such as green energy technologies and reducing reliance on fossil fuels, our society has made significant progress. However, there is still much work to be done. The average home presents opportunities for countless improvements in energy conservation. In particular, heating and cooling of homes present the opportunity for many inefficiencies to be addressed, including the lack of automation of residential climate control. For instance, to account for temperature variances, an individual may choose to run power-hungry central air systems as opposed to using manually adjusted power-saving alternatives. However, a sufficiently smart home automation system could replace human negligence and laziness and utilize alternative energy saving options that take advantage of the existing features of homes. By specifically targeting the windows of a home and envisioning them to become more active, we can ensure that the next generation of engineers that tackle this problem can build upon the foundations that we have created, allowing for true automation of this system.

A. Significance

As climate change becomes ever more relevant throughout the world, we foresee this solution carrying great weight in the future of the automation of homes throughout the world. Indeed, a home loses a great deal of its energy from non-insulated windows, and by not having an automated system that can adjust the window as necessary to let heat in or out, a great deal of unnecessary energy is wasted in the process of heating the home. In addition, by allowing for a simpler user experience via both an android application and a local manual override switch, it ensures that users will not need to strain themselves during the operation of the window, while also ensuring that they can control these windows irrespective of if they are in their home or not.

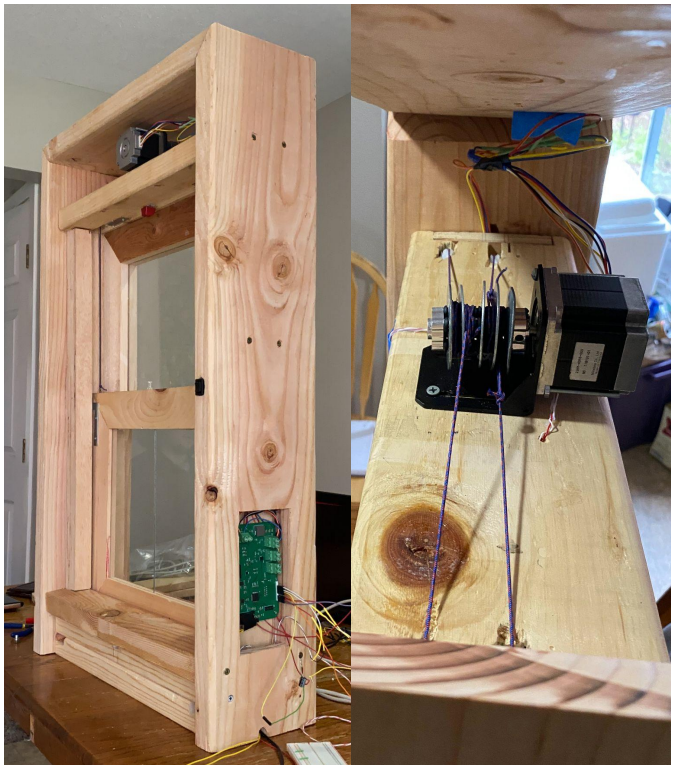

Figure 1: Our Model window to demonstrate our system with dimensions 36" tall, 19" wide, 7" deep.

B. Context and Competing Solutions in Marketplace

There are many companies that have attempted window automation to combat climate change, however there are not many that have tackled the action of actually opening and closing a window. One solution developed by Kintrol aimed at automating a custom designed window that is most likely found in commercial buildings [3]. They focused on specifically awning type windows which swing open from the bottom as well as jalousie windows which are better explained as a series of smaller awning windows built into one. Their system comes with a software component that allows for remote control and monitoring of the system. Other companies

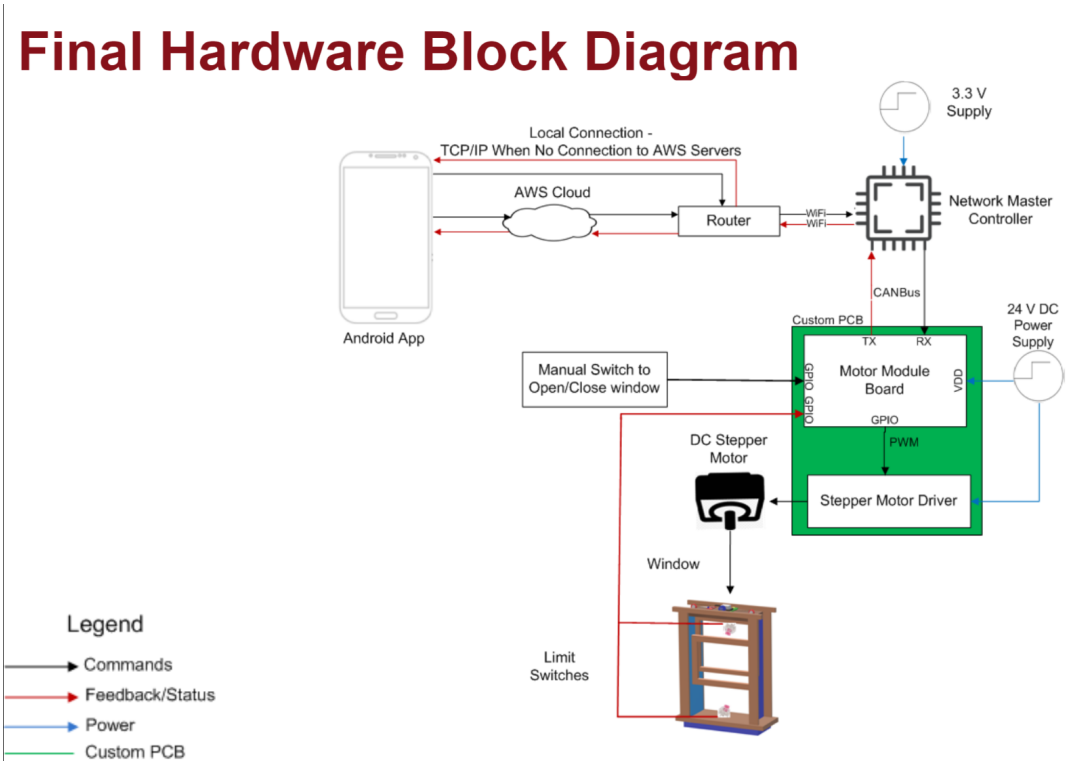

Figure 2: Block Diagram

have focused on the automation of window shades to intelligently control how much sunlight is let into a building. One example is lutron's Serena shade systems which integrates with their existing home automation platform which also integrates with home lighting [1]. Another solution on the market is a device called sol-Lux which is an automated retractable shade that operates solely from the sun [2]. This device is a great solution for combatting energy efficiency however it doesn't include any integration for data monitoring or user control its operation is reliant on the sun alone.

C. Societal Impacts

As development of this project began during the design phase, it became apparent to everyone involved that nearly everyone in the future who would purchase the next generation of autonomous homes would benefit greatly from our project. Simply by ensuring that homes become more Active and Intelligent, we can ensure that less and less unnecessary energy is lost during the average operation of a home, while also ensuring a user-friendly experience with the system that we have developed.

D. System Requirements and Specifications

At a high-level, this team envisioned and created four distinct subsystems in order to create our Active Window system. We envision that a user can use an Android Application in order to send commands to any window in their home. To facilitate communication to this Active Window system, the Android Phone Application can make use of the cloud (Specifically, Amazon Web Services IoT Core functionality) which not only handles authentication, but also allows for remote activation of the window from anywhere in the world. These commands are forwarded to a local Network Master Controller, which parses through the user's command and directs it to a specific window in the home. The Motor Module receives this command from the Network Master Controller, actuates a stepper motor on the test window, and sends a status update back through the whole system, which the user then receives on their application.

Therefore, the Active Window subsystems consist of the Network Master Controller, IoT/Phone Application, and the Motor Module that meet the following specifications:

- Capable of receiving signals from Amazon Web Services (AWS) for direct communication between the user and the rest of the system.
- Capable of transmitting commands from the Android Application through AWS to the Network Master Controller that drives a stepper motor driver, which in turn adjusts a window.
- User interaction via an Android Phone Application that transmits data via AWS to the controllers.
- User interface to control and monitor window status. The Motor Module's size is less than 14cm x 8cm in order to maximize installation in a home.
- DC Stepper Motor and Driver, utilize a 24V power

Final Software Block Diagram

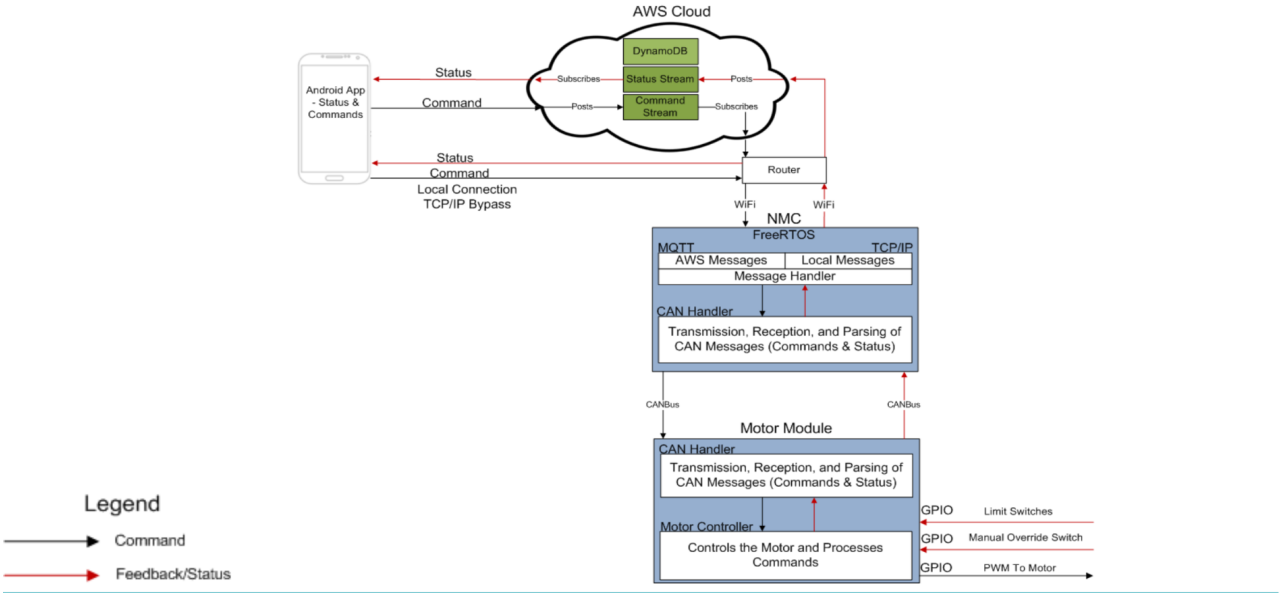

Figure 3: Software Block Diagram of the Active Window

supply and use a pulley system solution that can adjust the window via commands. This pulley system physically lowers and raises the test window once it receives the proper command.

- In addition, there is also a manual override switch for the test window that is capable of bypassing the Android Application and will raise or lower the window when it is flipped.
- In the event of external network failure, the Android Application is capable of sending commands to the Network Master Controller if they are on the same network.

II. Design

A. Overview

The active window is made up of four main components with the custom fabricated PCB being one of these four. The other three components include the network master controller, AWS IoT core, and an android application.

The custom PCB that the team designed within the requirements of the Manhattan2 project served as the motor module which was used to control the window from receiving commands remotely from the NMC and locally from a manual switch.

B. Model Test Window

The team designed a model test window to fully test the designed system in a real world environment. To do this, the stepper motor was attached via a pulley solution to the movable part of the window, and used a spindle to either wind or unwind depending on whether the window was moving up or down. Of note is that at the top and bottom of the window two limit switches were used in order to calibrate where the top and bottom of the window were located. On startup, the motor module moves the motor until it hits the top switch, and then does the opposite to hit the bottom switch to determine the bottom of the window. This design makes the window extensible to other window designs so long as they have these two limit switches. We refer to this automatic calibration as the homing routine.

C. Motor Module

The motor module became the focal point of our project in that it was the custom pcb we designed. This device needed to abide by many requirements to be compatible with the future work of the Manhattan2 system. The Motor module's basic function was to receive commands from the NMC to control the motor, to do this we used a A4988 motor driver which received the signals sent through CANBus from the XMC-4200 chip in the form of PWM that drove the motor. Also on the PCB there were two CAN transceivers of which we only used one, as well as a power Converter that could receive an input of 48V, despite our group only needing 24V. The reason there are two was to aid in the future development of the board in the Manhattan2 project.

D. Network Master Controller

The network master controller serves as the gate between the local network and the internet. This device is used to manage communication between devices within the network by transmitting data through CAN and connects to the internet through wifi. The device used to achieve this functionality is

the XMC-4800 evaluation board which includes a wifi receiver, and can transceiver which we made use of for our design.

E. AWS IoT Core

For the Manhattan2 project a local server will be placed at a home to handle the remote communication and data analysis of the system however since developing a server was outside the scope of our project we decided to make use of AWS IoT tools to act as our server and allow for the remote operation of our system. Inside AWS we made use of topics which are publish and subscribe message streams, publishers send messages and subscribers get messages. We also made use of a database hosted by AWS called DynamoDB which we used to store the latest status updates of the system so when the app boots up it can see the latest status of the system without needing to involve the local system. Finally, we also used Cognito for Identity Management between the AWS servers and the Android Application / Network Master Controller. This is used in order to identify who is communicating with the system. This adds a fortified secure way of communication with the final system.

F. Android Application

The Android Phone Application acts as the user interface to the entire system. On startup, a user can immediately see if they are connected to the Amazon Web Services (AWS) servers. From here, the user can make the decision whether to communicate via the AWS protocol or a local TCP/IP Protocol. Regardless of which option the user chooses, they will be brought to an interface which informs them which protocol they are using, and greets them with a slider to allow them to send a message to the specific window. On startup of either application, the user immediately sends a request to the Network Master Controller (NMC) to get a status update on what the most recent command received to the window was. The NMC sends an acknowledgement of the request, and returns the most recent status. Once the status is returned, a text string appears to the user, in addition to a 3D model representing the state that the window is in. The user can then send a command to adjust the window in steps of 10%.

If the user decides to utilize the TCP/IP protocol for local connectivity to the system, the application and the NMC need to be on the same network. Automatically, the NMC prefers communication via the internet, but once it loses connection to AWS servers, it begins waiting for a socket to open up on the Android Application. Once the socket is open, the two initiate the TCP/IP Handshake Protocol to transfer commands and send status updates to one another. Once internet connection is reestablished, the NMC publishes a message to AWS saying that it has reestablished connection and is waiting for new commands from a user. In addition, the NMC sends a command to the user over TCP/IP to tell them that they have reestablished connection to AWS and that the socket has been closed.

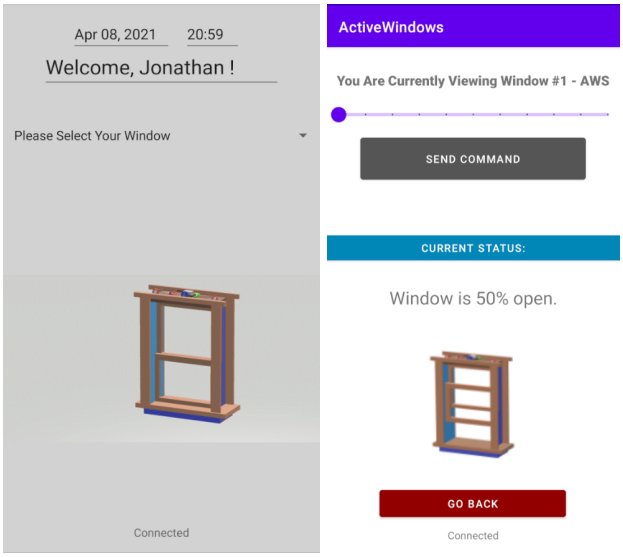

Figure 4: Application User Interface

III. The Refined Prototype

A. Prototype Overview

The final prototype of the system is consistent and in line with the system specifications laid out earlier in this document. Succinctly, the model window was fully constructed with the stepper motor mounted on the top of the window. The stepper motor's axle then had a spindle mounted on it with the final pulley solution in place. The limit switches, as discussed earlier, were mounted on the top and bottom of the window for the homing routine. There also exists a manual override switch on the right side of the window to actuate the window up and down. This manual override switch also has 2 1 uF capacitors in parallel on the line in order to reduce noise on the line. The finalized Motor Module PCBA was then attached to the side of the window in a small compartment. A CANBus wire and a power line then leave the window to be hooked up to a 24V power supply and XMC-4800 Evaluation Board, respectively. The XMC-4800 Evaluation board (The NMC) is then connected via WiFi to the router in the home to connect to either AWS or to be prepared for a TCP/IP protocol. Finally, the Android Application can be practically anywhere if it wants to connect via AWS, or on the same network if the user would like to communicate via TCP/IP.

In terms of a user interacting with the system, a user boots up the Android Application and selects to communicate via AWS or TCP/IP with the system. In either case, on startup the Android Application sends an immediate message to the NMC requesting for the current status of the window. The NMC responds with this status, and the application displays a 3D model of what percentage the window is open. The user can then send a command in 10 step intervals (0% open to 100% open) to the window. The NMC responds by forwarding this command to the Motor Module, the Motor Module actuates this command, and sends a "Operation Complete" message to the NMC, which then forwards this new status to the application, either via AWS or TCP/IP.

B. List of Hardware and Software

Software: FreeRTOS, Android Studio, DAVE IDE, Altium Designer, Texas Instruments WEBENCH, Amazon Web Services

Hardware: XMC-4800 board, 24V Stepper Motor, XMC-4200 board, A4988 Motor Driver, LMR33620ADDAR Power Converter IC, Power Supply, CANBus Wire, Window Hardware

C. Custom Hardware

The hardware used for the Motor Module included the XMC-4200 Microcontroller, the A4988 Motor Driver, The Stepper Motor, and a breakout breadboard. The design of the Motor Module required the use of each hardware components datasheet, which mapped the creation of a wiring diagram schematic of the module. Using this wiring diagram, allowed for a smooth process of physically wiring and connecting the each Motor Module component correctly. Once the module parts were fully assembled and connected, the system was tested using a USB connection between the XMC-4200 Microncontroller and DAVE IDE on a laptop. Using DAVE IDE Applications and C programming language the directions and speed of the Stepper motor module were able to be controlled.

The Motor Module's PCBA design process began with creating a schematic on Altium Designer. Planning out this schematic included figuring out the parts and functionalities the PCBA board would need to replace the wiring connections between the XMC-4200 Microcontroller board and the A4988 Motor Driver connected to a breakout board. Therefore the schematic and PCBA board were designed to have the functionalities needed from the microcontroller and the motor driver, as well as being capable of receiving CANBus commands and power conversion. The schematic can be seen below and is broken down into four parts, which were, the XMC-4200 Integrated circuit and oscillating clocks, the A4988 motor driver integrated circuit, the CANBus integrated circuits and connectors, and lastly the power converter utilizing a LMR33620ADDAR integrated circuit. After the parts from the schematic were organized and fully routed on the PCBA board, there were many edits and critiques made as a team before the design was finalized. Once the PCBA was finalized, the board was ordered and shipped through the JLCpcb website. The next step was populating the PCB with the necessary parts by using the soldering equipment and guidance available in M5. Both surface mount and through hole techniques were learned and utilized during the population process of the PCBA. After fully soldering the PCBA the team began testing using an XMC-4000 Debugger, and successfully tested and programmed every functionality of the board that was needed and implemented the board into the final Window design system. As planned, the PCBA ended up replacing the XMC-4200 Evaluation Board, the A4988 Motor Driver breakout board within our system.

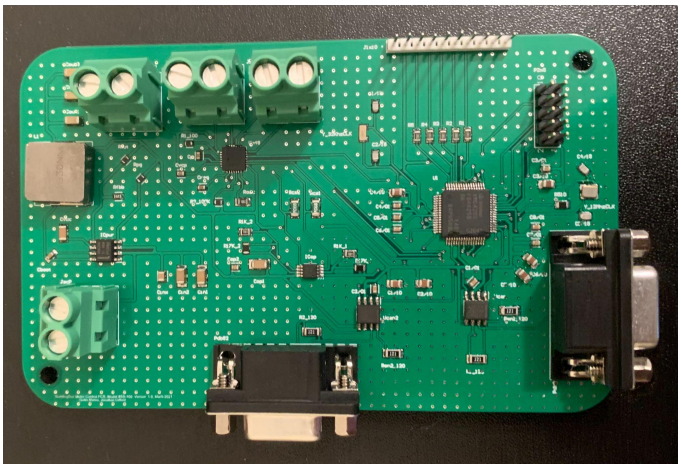

Figure 6: Custom Populated Motor Module PCBA

D. Prototype Functionality

As of the completion of the project post-FPR, the entire system works as laid out in the system specifications. Indeed, the entirety of the PCB is fully functional towards our

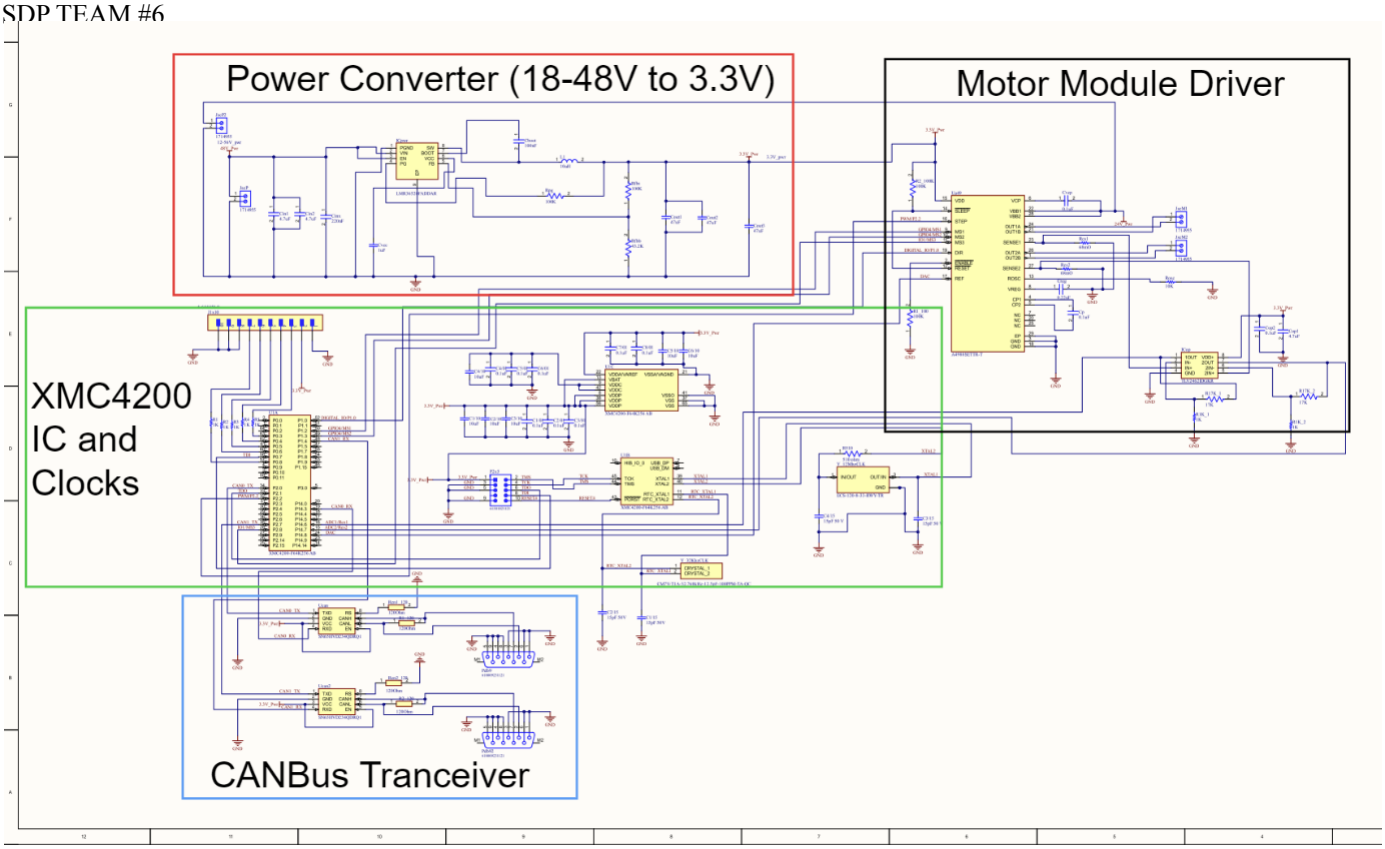

Figure 5: Custom Motor Module PCB Schematic

design and is capable of actuating the DC Stepper Motor, sending and receiving data via the CANBus transceivers. In addition, the motor module board also steps down any voltage between 18 - 48 V down to 3.3 V for use in powering the digital components (XMC-4200 and CANBus transceivers). Of note, however, is that the CANBus transceivers that the team used deviate slightly from convention, these two are powered by 3.3 V as opposed to the more typical and more widely available 5 V transceivers. This was done in order to simplify the power converter module to have only one step, down to 3.3 V. This had no practical effect on the usability of the CANBus transceivers.

E. Prototype Performance

The prototype window and system performed as intended. When connected to AWS, the NMC is able to communicate with the Android Application, both for status updates and commands. The Motor Module is able to respond to forwarded messages from the NMC, both for status updates and for actuation of the window, which it does without issue. Additionally, it is able to communicate with NMC for updates of its status. The manual override switch also works as intended to move the window up and down, as well as updates its status after such movements. Additionally, our TCP/IP local connection works as intended, and provides all of the same functionality as the AWS mode of operation.

IV. Conclusion

In conclusion, the team is extremely satisfied with what was accomplished with this project. Being able to complete multiple distinct subsystems with the added complexity of the ongoing COVID-19 Pandemic made this project unique as compared to previous years. However, what can be said is that being remote, it allowed for an impressive demonstration of the working system. As an example, Jonathan Clifford was over 85 miles away from where the window was set up, and was shown to actuate the window to whatever percentage that he wanted it to be. In addition, making a simple test website that any user can control the system also allowed for users to have an impact on the system regardless of their own location, showcasing a primary aspect of this project - being able to control your home regardless of where you are via the cloud.

Ultimately, what was accomplished with this project, especially given the circumstances, is something that the entire team is greatly proud of, given the scope and complexity of the project. It also is a point of pride for this team that future engineers will be able to build off of this project to design the next generation of smart homes.

Acknowledgments

This group would like to especially thank Professor Arman Pouraghily, whose wise counsel, experience, and encouragement made this project what it is today. We would also like to thank Manhattan2 CTO Mr. Glenn Weinreb for bringing this project to our attention, and greatly assisting us during this project, his experience and insight were invaluable to the success of our project. We would also like to thank Loren Simpson and Joshua Towner for their assistance in calculating the maximum weight that our system could sustain.

References

[1] *Shading Solutions from Lutron Provide Energy Saving Light Control*. [Online].Available:https://www.lutron.com/en-US/Residential-Commercial-Solutions/Pages/Residential-Solutions/ShadingSolutions.aspx. [Accessed: 03-May-2021].

[2] "Smart Solar Window Awnings: Intelligent Design: Sol-Lux Window Shades, "*Sol*. [Online]. Available: https://www.sol-lux.com/smart-technology/. [Accessed: 03-May-2021].

[3] "HOME," *Kintrol*, 27-Oct-2020. [Online]. Available: https://kintrol.com.au/. [Accessed: 04-May-2021].

Appendix

A. Design Alternatives

In this project there were a great deal of systems to design. One of the components that required design was the window. Since the model was used to show the functionality of our system in a home, we made some designs to obey the window dimensions of a current residential home. The limitation was primarily the depth of the window, which had to be between 3.5 and 7 inches. Another limitation was the weight of the window, which needed to be as light as possible to lessen the load on the motor. One of the requirements of the Manhattan2 project was to use a 24V motor, the team did not do an extensive amount of research into the maximum amount of weight that this window could sustain. However, the team calculated the maximum amount of weight that the stepper motor can sustain, 9.64 kg. However, the team also determined that practically the window could actuate a maximum load of 6.75 kg. Further discussion regarding the calculations can be found in Appendix G.

Another potential design alternative that can be mentioned is using a different cloud service provider than Amazon Web Services. Ultimately, the choice of using AWS was due to the team's previous experience with AWS, a great deal of support for the service, and the extensibility of the service. For instance, AWS has access to Amazon Sagemaker, which is the service's Machine Learning tool. With the way that the team designed the system (Posting to DynamoDB, for instance) Sagemaker could learn a user's habits and automatically control the window for them. However, another potential service that the team could've used is Google's Firebase. Firebase is a service similar to AWS, however many of the tools work off of a database as opposed to AWS' message streams. This could've allowed for a more thorough history that the user could view for their window over time.

B. Technical Standards

- CANBus, ISO 11898-1, ISO 11898-2
- WiFi, IEEE 802.11
- PCB, FR4-Standard Tg 140C

C. Testing Methods

The overall system was thoroughly tested for functionality. In general, it was important to test for edge cases and system synchronicity. Because of the nature of our system being composed of three distinct subsystems, a major concern was maintaining a homogenous state throughout.

Most of the experimentations that were done on our system was to ensure the handling of each edge case and that each component works properly. Some of the edge cases included connection status messages in the app being updated properly, heartbeat messages sent between the phone application and the NMC to ensure the connection is still there, and commanded actuation while the window is being locally actuated. Work was also done on the model window wiring as well as attempting to reduce the vibrational noise from the stepper motor by adding rubber grommets between each metal contact for a better presentation on Demo Day.

D. Project Expenditures

Part	Price
XMC-4800 Evaluation Board	$95.93
XMC-4200 Evaluation Board	$55.47
24V 23KM-K066-00V Stepper Motor	$50.00
24V Shield (Motor Driver)	$25.97
A4988 Motor Driver	$8.00
XMC-4800 Processor	$23.65
XMC-4200 Processor	$14.62
XMC-4000 Debugger	$95.00
PCB Components	$22.89
Custom board (Version #1)	$58.15
Custom board (Version #2)	$35.11
Mouser Order For Crystal Oscillator	$14.79
DigiKey Order (Remaining PCB Parts)	$49.37
TOTAL	$548.95

E. Project Management

Since the project has a great deal of moving parts, each part of the project had to be broken up into smaller components that team members specialized in. The work was broken up as follows:

Jonathan Clifford - *Team Coordinator, Android Application Developer, & Amazon Web Services (AWS) Developer*

Frank Cremonini - *CANBus Communication Software Developer & Motor Module Code*

Griffin Manns - *Stepper Motor Module & Altium Designer PCB Lead*

Connor Moore - *Network Master Controller Developer, Budget Management, Model Window Designer & Fabricator*

F. Beyond the Classroom

Jonathan Clifford: As the Team Coordinator, I was in the unique position of being able to assist in the development of nearly every part of this project. Not only was I able to hone my project management skills, software development skills with the Android Application, and web development skills, but I also obtained experience being able to debug potential problems with our circuits and assist in the PCB design in

Altium. Indeed, the skills that I developed during this project will absolutely carry over into my professional career.

Frank Cremonini: In my time working on this project, I have developed many skills, both as a computer and electrical engineer. First and foremost, I extended my knowledge of CAN, both in protocol and in usage. Although I had previous experience working with it, building it from the ground up helped to develop my comfort level with it. Additionally, I have learned lots about creating a bare metal system that can handle the functionality of many tasks.while maintaining state and consistency. On the electrical engineering side, I have gained far more comfort in working with, measuring, and monitoring electrical systems. In particular, our project necessitated the use of many different parts that needed to be correctly connected so as not to break anything. When something was non-functional, diagnosing problems with electrical components gave me great experience in thinking from an electrical engineering perspective rather than my usual computer based point of view. Datasheets for our evaluation board and processor were full of very useful information, and reading through and locating key information was a great exercise. In general, this knowledge will serve to help me further succeed in my career.

Connor Moore: To help realize this project that we have made there were several things that we needed to learn. For my contributions to the team which included programming the NMC and building the model window, I had to first learn how to program a real time operating system and properly handle the management of tasks from within the FreeRTOS software. The FreeRTOS software is very well documented so it was very helpful referencing their API to help construct the program. Building the model window required a good amount of mechanical knowledge and structural analysis to achieve the intended functionality while maintaining a sound prototype. I also assisted with the android application development since it needed to interface directly with the NMC so myself and Jonathan worked together to bring this functionality to life. Learning new software is something that is guaranteed to come with a career in computer engineering and having the experience in learning one helps with learning future ones.

Griffin Manns: My responsibilities for this project included designing and wiring our Stepper Motor Module, as well as designing and putting together our custom PCB for the Motor Module. While creating our Stepper Motor Module I had to do research on how our specific stepper motor and stepper motor driver worked and wired together with an XMC-4200 microcontroller, in order to drive and control the motor's movements. In order to wire the module, I mainly utilized online stepper motor tutorials, as well as the datasheets for our stepper motor, motor driver, and microcontroller. Once the system was wired I then utilized DAVE IDE resources and Applications as well as help from my teammate Frank, in order to program the module and control the basic movements of our stepper motor, using the C programming language. Designing the PCB required a great deal of research and time reading datasheets, finding parts, and learning how to navigate the Altium designer tools and functionalities. I seeked as much help as I could get from both the altium support group, as well as our Manhattan2 Contact, Glenn Weinreb, in order to complete the design of our PCB on Altium. Once the PCB design was finalized, I then utilized the soldering equipment at M5, and received soldering training from m5 staff, in order to fully assemble and solder our parts onto the PCB board. Overall, I had to use my communication, collaboration, and problem-solving skills in order to learn new design techniques and softwares needed to complete our project.

G. A Word On The Maximum Weight Our Design Can Sustain

Specifications for Stepper Motor and System:

- Holding Torque: .9 Nm
- Diameter of Stepper Motor Shaft: 6.35 mm
- Winding Shaft: 19.05 mm

$$Torque = Force \cdot Distance \tag{1}$$

The distance is the radius of the shaft. The shaft is the cylinder upon which the string winds.

$$Force = \frac{Torque}{Distance} \tag{2}$$

$$m_{max} = \frac{\frac{Torque}{Distance}}{g} \tag{3}$$

Here, g is the gravitational constant, and mass is the maximum weight that the stepper motor can hold.

$$9.64\text{kg} = \frac{\frac{.9}{.009525}}{9.8} \tag{4}$$

The stepper motor can only **hold** 9.64 kg. In order to raise the weight, conventionally only 70% of the holding torque is used. With that in mind, the maximum mass that the stepper motor could realistically lift would be:

$$m_{max} = 6.75\text{kg} \tag{5}$$

H. Source Code For Our System

All source code can be found in the following repositories:

A. Network Master Controller & Motor Module
   a. https://github.com/cmoore147/ActiveWindow
B. Android Application
   a. https://github.com/JonathanClifford/SDP2021_Team6_AndroidApplication
C. Website Source Code (Developed with Mobirise & created a smaller page to control the window with a website for demo day)
   a. https://github.com/JonathanClifford/sdp21ActiveWindowsWebsite

J. Relevant Datasheets For The System

There are a variety of datasheets present for the components of the completed system. Here, we will only list the most relevant to the final design that the team utilized.

A. XMC-4800 Evaluation Board Datasheet (Network Master Controller)
   a. https://www.infineon.com/dgdl/Infineon-Board_User_Manual_XMC4700_XMC4800_Relax_Kit_Series-UM-v01_02-EN.pdf?fileId=5546d46250cc1fdf01513f8e052d07fc
B. XMC-4200 Microprocessor Datasheet (Motor Module Processor)
   a. https://www.infineon.com/dgdl/Infineon-XMC4100_XMC4200_DS-DS-v01_04-EN.pdf?fileId=5546d462696dbf120169817056f938ff
C. LMR33620ADDAR Power Converter IC Datasheet (Power Converter)
   a. https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Flmr33620
D. A4988 Motor Driver (Motor Module Stepper Motor Driver)
   a. https://www.pololu.com/file/0J450/A4988.pdf
E. 24V 23KM-K066-00V Stepper Motor Datasheet (Stepper Motor)
   a. https://media.digikey.com/pdf/Data%20Sheets/NMB-MAT/23KM-K.pdf
F. SN65HVD234QDRQ1 IC (CANBus Transceiver)
   a. https://www.ti.com/lit/ds/symlink/sn65hvd234-q1.pdf?ts=1620042283513&ref_url=https%253A%252F%252Fwww.google.com%252F
G. Kailh Box Pink Switches (Limit Switches)
   a. https://cdn.shopify.com/s/files/1/3099/8088/files/BOX_Pinks.pdf?2934967025343846493