Active Windows

Michael Chan, EE; Dingbang Chen, CSE; Nathan Johnson, CSE; Tien Li Shen, CSE

Abstract— **Climate change is an often talked about problem that impacts everyone around the world. CO2 emission represents a large part of this issue and a solution is to build more efficient homes. There are a plethora of smart home systems that have differing functions and that can be costly to implement. Our system is about establishing an electrical and communication standard that can be implemented in newly built homes to cut costs and be more eco-friendly. This includes the development of a new CAN transceiver that would allow devices to be connected in a tree topology. A single window can expect to take inputs from sensors or other nodes on the wired network, and move windows and blinds based on sensor states, signal generated from other nodes on the network, or user commands from mobile apps or on-window touch displays.**

I. Introduction

BUILDING automation systems can reduce operation costs and carbon emissions by improving energy efficiency. However, many current solutions are manufacturer specific and expensive, making widespread adaptation difficult. Our project aims to assist the non-profit Manhattan 2 organization in developing electrical and communications standards that define how devices interconnect within the buildings of the future [1]. This entails the development of a new CAN transceiver circuit that supports tree topology wiring.

A. *Significance*

The consumption of energy continues to increase as the world population grows. From 1971 to 2014, the World's electricity consumption per capita rose 2.6 times, from 1200 kWh to 3132 kWh. To keep up with demand, the production of energy also rose at the cost of negatively impacting the environment, with 89.6% of this energy production coming from non-renewable sources [2]. This raises two long term issues. The first issue being global warming and the second being the accelerated depletion of coal, oil and natural gas reserves. To offset this negative impact, renewable energy is being used to supplement the demand. Additional methods of reducing greenhouse gasses comes from having energy efficient systems.

B. *Context and Competing Solutions in Marketplace*

Most of the home automation solutions today utilize wireless communications between devices and require monthly subscription fees for automated cloud services. Wireless communication has the drawback of consuming more power, highest packet loss, and unstable connection compared to the more traditional wired communications. Our project is completely open source and utilizes CAN bus wired communication protocol promising robust connectivity between devices and free software.

Active window system has a hierarchy for controlling devices, offering the ability to manage large numbers of devices with centralized control units. Master Controller that manages multiple Network Controllers, each Network Controller manages multiple Devices, and each Device optionally manages multiple Subnetwork Devices. Breaking this up into many different networks means that if one wire breaks, the entire system is not affected. The goal of the design is to provide reliability, fault tolerance, and quality.

C. *Societal Impacts*

Our project aims to demonstrate that a low cost, reliable home automation system is possible to create. If this project were to be expanded for use by the general public, it would create a non-negligible impact on greenhouse gas emissions by residential buildings. This system would have a much greater impact on society than existing solutions because it would be possible to widely adopt due to its cheapness and simplicity. The overall societal goal of this project is to reduce greenhouse gas emissions to a manageable level where climate change is no longer an existential threat.

D. *System Requirements and Specifications*

Requirement	**Specification**	**Value**
Transceiver	Power	<5mA of power
	Logic	Defines logic 1 and logic 0
	Canbus	Supported
Networking Software	Topology	Tree topology
	Packet Addressing	Point-to-point and broadcast
	Reliability	No dropped packets under nominal operating conditions

Table 1: Requirements and Specifications

II. Design

A. *Overview*

Active Window is a home automation system focused on connected smart windows. A single smart window can have configurable electronic and mechanical components such as displays, sensors, motorized windows, and motorized blinds. All of the windows are connected via CAN bus wired communication in tree topology wiring promising robust control and connectivity.

Our part of the Active Window Initiative is the development of custom CAN transceiver and networking software framework. Additionally, we implement motors and sensors to demonstrate a complete input/output system with communication across the network.

B. Stepper Motor Assembly

The reason why we choose stepper motor as our output is because stepper motor has many advantages for us to use: precise step control, controllable speed adjustment and excellent holding torque. In our project, we used a 28BYJ-48 stepper motor with ULN2003 control board to control the mock window we built. 28BYJ-48 stepper motor [3] is a four-phase eight-beat motor with a voltage of DC5V—DC12V. When we apply a series of continuous control pulses to the stepper motor, it can continuously rotate. Each pulse signal corresponds to a change in the energization state of a phase or two-phase winding of the stepper motor, which corresponds to the rotor turning a certain angle (a step angle). When the energized state changes to complete a cycle, the rotor will rotate through a tooth pitch. The four-phase stepper motor can run in different power-on modes. In this project, we did not use any existing project or header file to control the stepper motor, but by connecting the 4 pins of the stepper motor to the XMC4200 Platform2Go development board [4] and set them as outputs. Accurate control of the stepping motor is achieved through the eight-beat drive. At the same time, we can control the angular displacement by controlling the number of pulses, so as to achieve the purpose of accurate positioning; at the same time, you can control the speed and acceleration of the motor rotation by controlling the pulse frequency, so as to achieve the purpose of speed regulation. Not only that, when the stepper motor is energized but not rotating, the stator will lock the torque of the rotor. This is very practical on our mock window and avoids potential dangers.

C. Sensor Assembly

The system consists of 3 sensors types: optical sensor, water level sensor, and temperature sensor. The optical sensor used in our system is the 02-LDR2 photoresistor manufactured by NTE Electronics. The sensor is connected in a voltage divider and the circuit is supplied with 3.3 V, *see Figure 1*. A probe from the microcontroller reads the data with ADC.

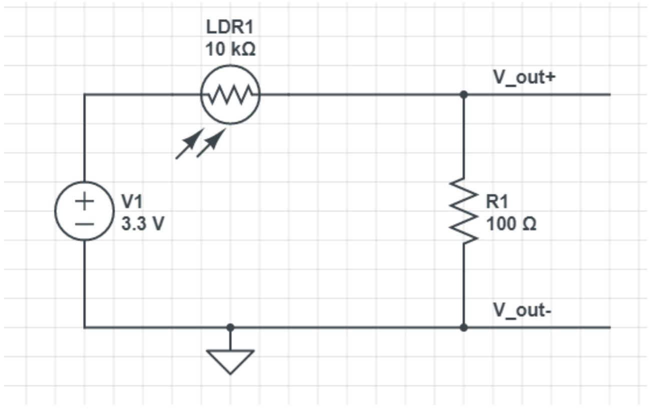


Figure 1: Photoresistor voltage divider circuit.

The water level sensor, *see Figure 2*, is supplied with 3.3 V and the output connects directly to the ADC port on the microcontroller.

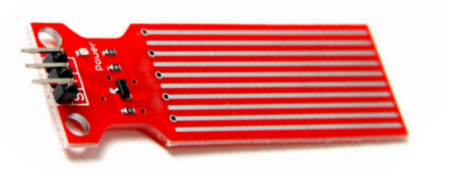


Figure 2: Water level sensor used to operate windows when it rains.

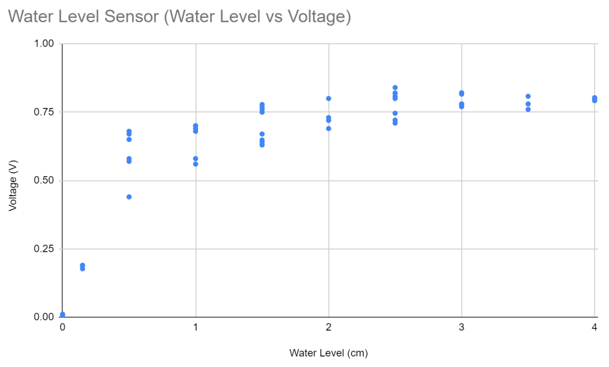


Figure 3: Water level sensor voltage behavior. The voltage data is collected by reading the registers in the ADC.

In *Figure 3*, the voltage behavior of the water level sensor is generally logarithmic with significant standard deviation in voltage at each water level. Water level as low as 1.5 cm could be reading the same voltage as the max of 4 cm.

The temperature sensor used in our system is the TMP36 by Analog Devices [5], *see Figure 4*. The sensor is supplied with 3.3V and the output is directly connected to the ADC port on the microcontroller. We test our sensor against commercially available sensors to check that our sensor is correctly calibrated.

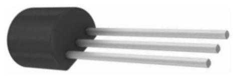


Figure 4: TMP36 Temperature Sensor

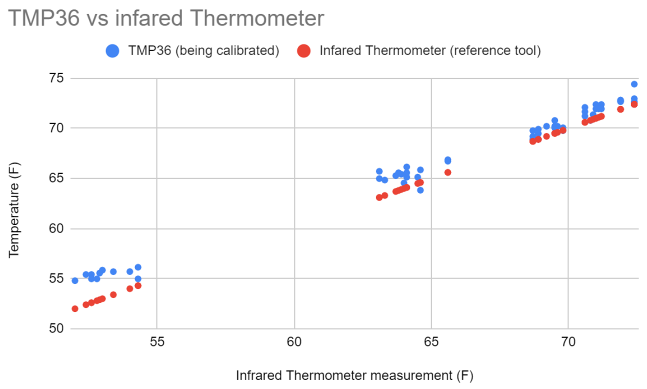


Figure 5: TMP36 vs reference thermometer.

The data in *Figure 5* is taken by pointing the infrared thermometer at the TMP36 and taking a measurement. The TMP36 mostly reads 2 degrees F higher than the reference thermometer. We calibrated the TMP36 to match the temperature reading of our reference thermometer.

D. *Processor Board*

In our distributed system, there are multiple processing boards that handle network communication, sensor data acquisition, and stepper motor actuation. Each board is an off-the-shelf development platform based on the Infineon XMC-4200 chipset, *see Figure 6*. We chose this particular microcontroller for two primary reasons. The primary purpose was to make our work compatible with existing code written for the same product by other Manhattan 2 developers. Likewise, we wanted our system to be easily extendable by future developers and researchers in the group. As it turned out, we didn't end up using any previous code for the board and wrote everything from scratch.

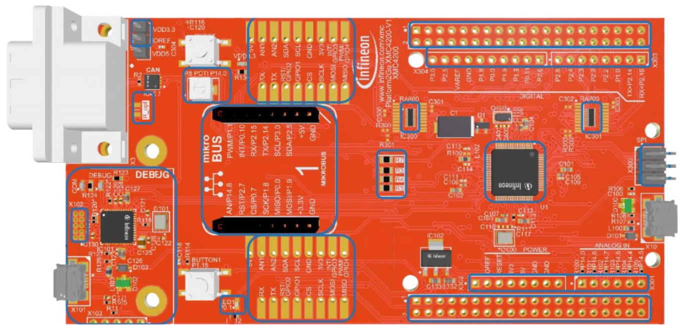


Figure 6: XMC-4200 Platform2Go development board

The second reason for using this development platform was because of the microcontroller's particular CANBus hardware features. CANBus is not as commonly used for inter-microcontroller communication as other serial protocols such as I2C, UART, and SPI. Due to this, not many microcontrollers have hardware support for CANBus communication. Of course, we could've used an ordinary microcontroller and manually programmed it to handle CANBus communication, but the XMC-4200 already implemented the CANBus link layer in hardware which reduced effort on our end. Additionally, because the XMC-4200 could handle CAN messages in hardware, it meant the CPU was free to handle our other application code without a complex interrupt driven program. The development board itself also included a CAN transceiver for driving the signals out onto the bus.

Overall, the board was an incredibly useful tool for getting started with this project. We chose to continue utilizing it for our final system because our custom PCB was exclusively focused on replicating the functionality of the CAN transceiver.

E. *Physical Layer Custom CANBus Transceiver*

A new means of communication between electrical nodes were needed to meet the requirements of Table 1. A transceiver was needed to be designed to connect to a CANbus transceiver to allow devices in a network to be wired in a tree pattern. The transceiver was modeled using the TINA program [6] and then built on a breadboard for prototyping, *see Figure 7* for two transceivers on a breadboard.

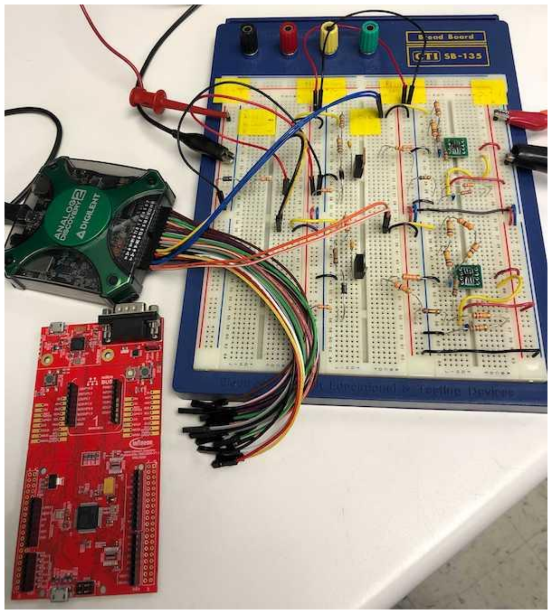


Figure 7: Breadboard with two transceivers

A Master Controller Power supply is used to establish a voltage difference of 16 Volts between the Data+ and Data- wires in this system. A voltage of 16 was chosen to allow for a large change in values to make a distinct logic 0 and logic 1 reading and also allow for longer lengths of wires to be used due to expected voltage drops. This system takes input of 0 -

3.3 volts to cycle a NPN transistor on and off. The transistor will short the Data+ to Data- allowing for messages to be sent from the transmitter end, *see Figure 8*. The receiver is powered by 3.3 volts and takes the input from the difference between Data+ and Data- wires. Using a comparator, the output between the inputs return 0-3.3 volts.

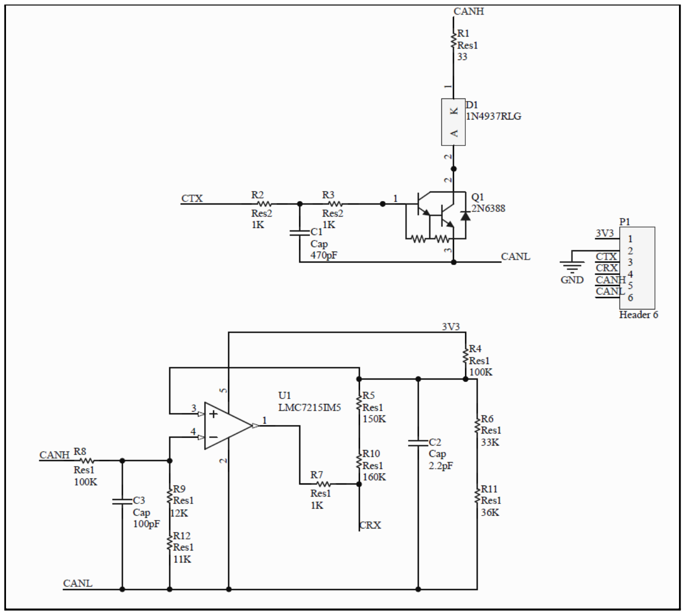


Figure 8: Transceiver schematic

To connect the Transceiver to the CANbus TX and RX, inverters were needed due to the CANbus at steady state reading 3.3 volts. Without the inverters, CAN TX would continuously short Data+ to Data-, *see Figure 9*.

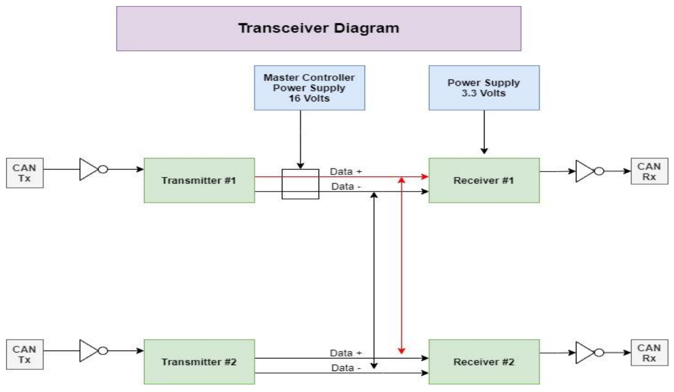


Figure 9: Transceiver integration to XMC4200 CANbus connection

F. Model Window Assembly

To demonstrate actual reactions to input from the system's sensors, we use multiple stepper motor assemblies that actuate two small model windows. These model windows are approximately six inches tall by four inches wide, and are made out of a balsa wood frame with plexiglass windows and a cloth blind. Both the blinds and windows themselves are fully operational, and can be raised/lowered using stepper motors, *see Figure 10*. The windows are taped with a green "X" for higher visibility of window operation.

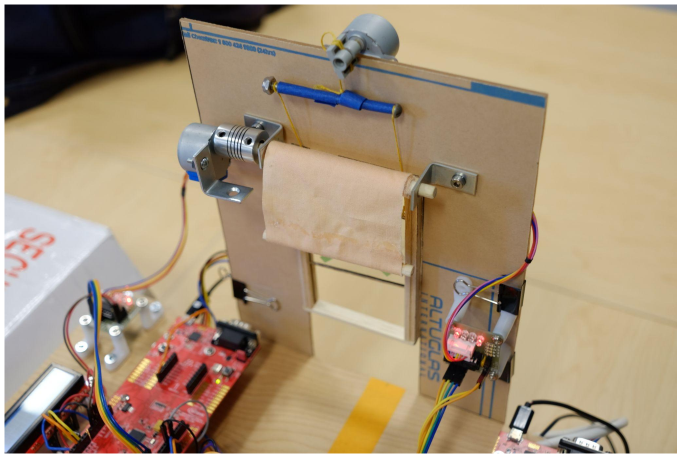


Figure 10: Model window assembly

G. Microcontroller Software System

Our project was somewhat challenging to program because it is a distributed computing system, rather than just a single monolithic computer. This raised challenges because some boards needed to perform slightly differently depending on their position in the CAN network and different peripherals attached. Additionally, it is imperative for the separate microcontrollers to maintain a coherent global state in order to operate correctly. For example, in our system the two model windows are operated by separate microcontrollers, but both windows must both have the same state (up or down) at any given time. This requires careful bookkeeping and communication between any microcontrollers involved in the operation of the window, in order to avoid any situation where the microcontrollers get out of sync.

Another challenge for this system was creating a program that could react quickly to stimulus. Stimulus in this case could either be CANBus messages from other nodes, or from changes in an observed sensor state. This was achieved through an interrupt driven program, versus a polling based system, *see the block diagram in Figure 11*. Interrupts were used to avoid using extra CPU time to check for stimulus. Two interrupts were used to drive the system: a timer triggered interrupt that tells the system to read a fresh batch of sensor data, and a network triggered interrupt that fires when the board receives a new CANBus message. The timer triggered interrupt fired every 10 milliseconds, which for the purposes of our system was an inconspicuous amount of delay.

In our project, there is one board that acts as a repeater between two separate CANBus electrical networks. This effectively makes the entire distributed system appear as if it is on one single electrical bus. We chose to do this to demonstrate that our system supports a flexible network topology.

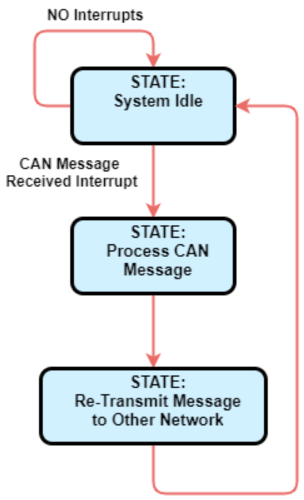

NO Interrupts
STATE: System Idle
CAN Message Received Interrupt
STATE: Process CAN Message
STATE: Re-Transmit Message to Other Network

Figure 11: State diagram for microcontroller acting as a repeater.

The other type of microcontroller in our project consisted of attached sensors and motors. In this case, the system needed to react to incoming CANBus messages and also collect sensor data periodically, *see the block diagram in Figure 12*.

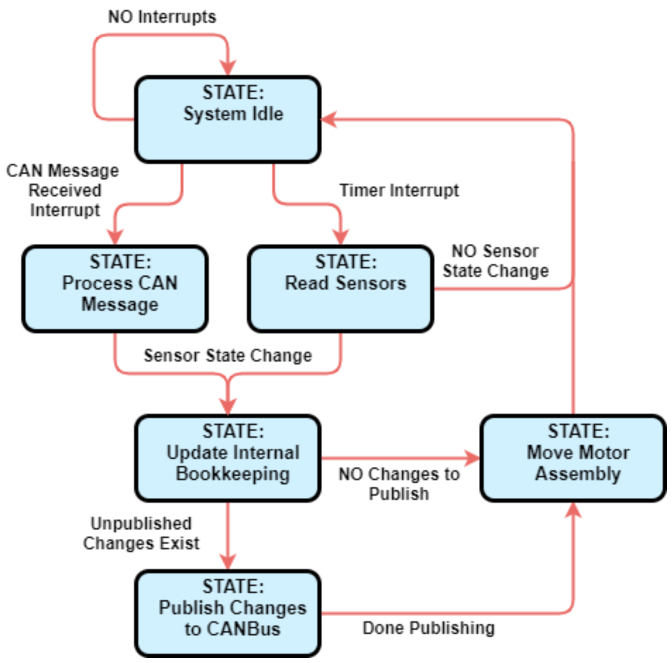

NO Interrupts
STATE: System Idle
CAN Message Received Interrupt
Timer Interrupt
STATE: Process CAN Message
STATE: Read Sensors
NO Sensor State Change
Sensor State Change
STATE: Update Internal Bookkeeping
NO Changes to Publish
STATE: Move Motor Assembly
Unpublished Changes Exist
STATE: Publish Changes to CANBus
Done Publishing

Figure 12: State diagram for microcontroller with motor and sensor peripherals

III. The Refined Prototype

A. Prototype Overview

This prototype is divided into two networks, left and right, *see Figure 13*. Each network is supplied with 16 volts and is housed in a grey electrical box. Each XMC4200 board is connected to a PCB Transceiver. The XMC4200 board on the bottom center has two PCB's to allow for message forwarding between networks. Each network has three sensors and two motors to drive the operation of the windows and blinds.

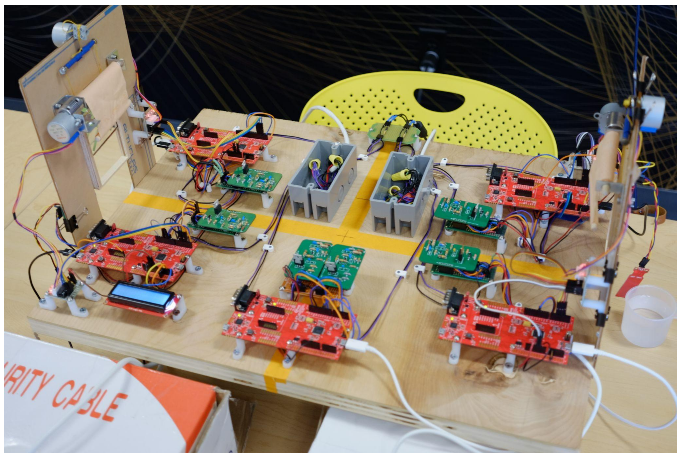


Figure 13: Prototype with model windows

B. List of Hardware and Software

- DAVE IDE (Eclipse based IDE for programing and debugging Infineon embedded systems)
- TINA (Toolkit for Interactive Network Analysis - Circuit Simulator)
- Altium (PCB design tool)

C. Custom Hardware

Our custom hardware was the transceiver and the PCBA was designed with Altium, *see Figure 14*. This PCB was attached to a protoboard that included an inverter.

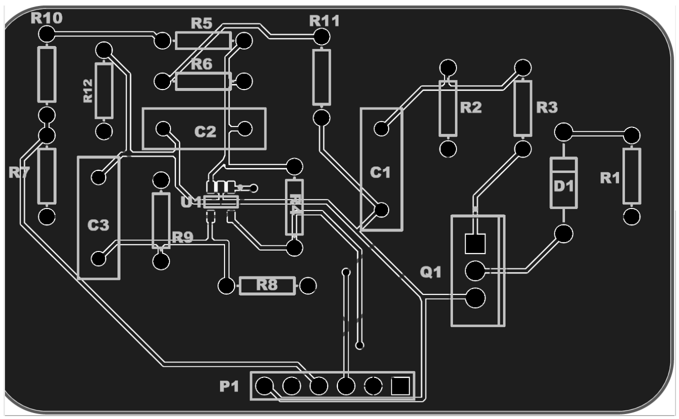


Figure 14: PCB Transceiver from Altium

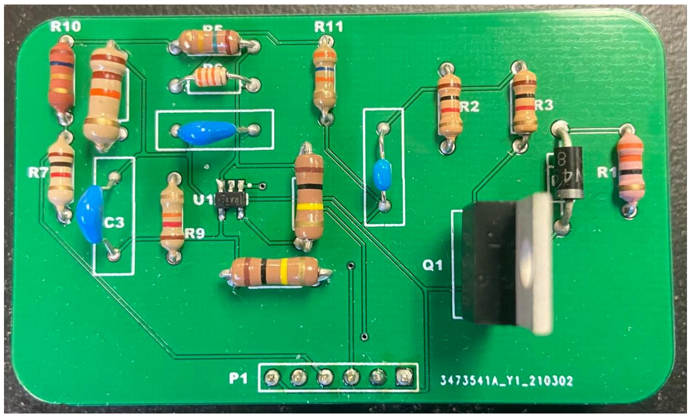


Figure 15: PCB Transceiver with components solder

This custom transceiver can be made smaller by changing out through-hole components for SMT parts. The custom transceiver can also be improved by adding two inverters onto the PCB. In our current prototype, the PCB attaches to another protoboard that has the inverters, see *Figure 16 and Figure 17*.

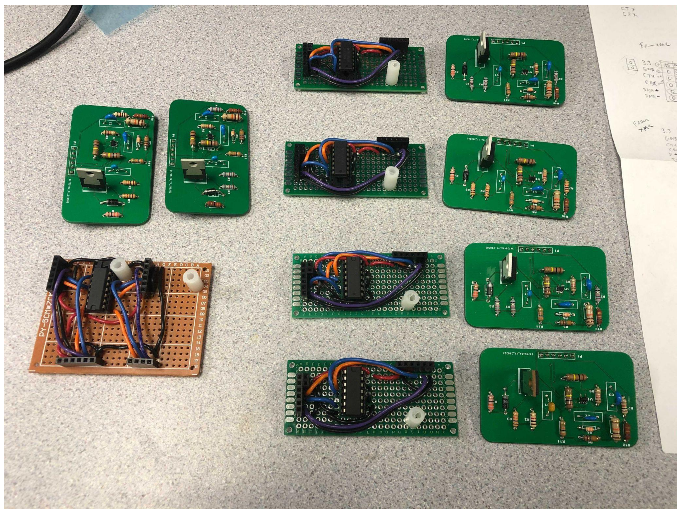


Figure 16: Custom Transceiver and Protoboards with inverters

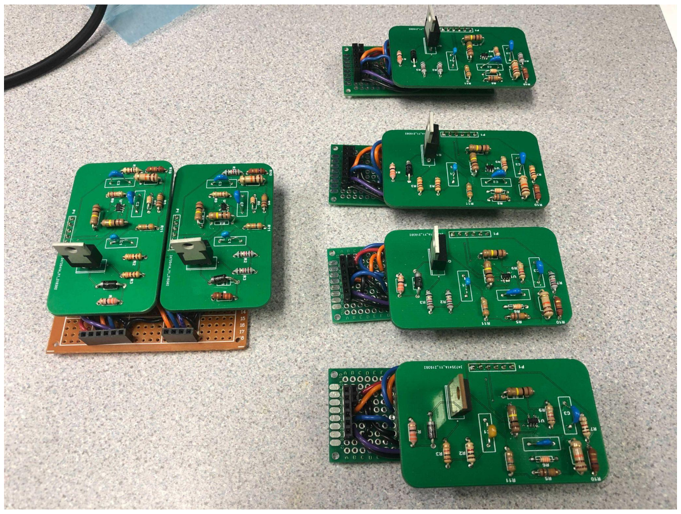


Figure 17: PCB attached on Protoboards

D. *Prototype Functionality*

Prior to full system operation, our team had issues integrating the transceiver with the XMC4200 development board. There was a mismatch between what the TX pin from the XMC4200 output, to the input of the transmitter. At steady-state, the TX pin is held at 3.3 volts. That 3.3 volts is also the value that actuates the transistor, allowing the data+ voltage of 16-volts to be shorted to data-, approximately 2.5 volts. We wanted steady-state voltage on the TX pin to not actuate the transistor on the PCB. To correct the issue, we connected inverters to both the TX pins and RX pins as seen in *Figure 9*.

Another issue we encountered was testing the PCB after soldering all the components. Using an Oscilloscope and a voltage meter, we determined that voltage was not present at the transistor. After verifying each component on the PCB was soldered correctly, we found that the stenciling for the diode was backwards for D1, see *Figure 14*. This prevented voltage from reaching the transistor. To correct this issue, we desoldered and flipped the diode to face the correct direction, see *Figure 15*.

Our prototype was fully functional. The system took inputs from 3 different sensors (Temperature, light, and water level). When a threshold was reached, data was broadcasted across the 16-volt networks to activate an output (motor driver).

E. *Prototype Performance*

Requirement	**Specification**	**Value**	**Goal Status**
Transceiver	Power	<5mA of power	Not met
	Logic	Defines logic 1 and logic 0	Achieved
	Canbus	Supported	Achieved
Networking Software	Topology	Tree topology	Achieved
	Packet Addressing	Point-to-point and broadcast	Partially met (no point-to-point)
	Reliability	No dropped packets under nominal operating conditions	Achieved

Table 2: Status of requirements

We were able to meet all of the requirements with the exception of determining transceiver power and implementing point-to-point packet addressing.

The power of our system was difficult to determine because the XMC4200 development boards that we used already were drawing 10mA during operation. These boards help with the programming and the connection of the hardware, but overall draw more power than we need.

Our system was able to broadcast information across all nodes, but we were unable to implement point-to-point packet addressing.

IV. Conclusion

Our final system was able to take stimulus from three different types of sensors (Temperature, optical, and water level) and transmit action to our motors for blind and window operation. We were able to include multiple states for our window blinds that function depending on the voltage reading of the light sensor. The temperature and water level sensors were able to close and open our custom built windows. Our custom PCB transceiver was able to send data between all nodes and between two networks for a maximum distance of 1000 feet.

Acknowledgment

We would like to thank our advisor, Professor Yao, for the continued guidance and support during this project. We would also like to thank Professor Jackson and Professor Tessier for the critical feedback and recommendations during our presentations. Also, we would like to thank Glenn Weinreb for giving us the opportunity to work with the Manhattan 2 development initiative and providing us with resources.

References

[1] Mobilize to Innovate to Reduce CO2, Manhattan 2, https://www.manhattan2.org/

[2] Chioran D, Valean H. Design and Performance Evaluation of a Home Energy Management System for Power Saving. *Energies*. 2021; 14(6):1668. https://doi.org/10.3390/en14061668

[3] Kiatronics 28BYJ-48 stepper motor, Data Sheet, *mouser,* https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf

[4] XMC4200 Platform2go, Data Sheet, *Infineon,* https://www.infineon.com/dgdl/Infineon-XMC4200_Platform2Go-UserManual-v01_00-EN.pdf

[5] TMP35/TMP36/TMP37 Data Sheet, *Analog Devices* https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

[6] TINA-TI Simulation tool, User Manual, *Texas Instruments incorporated,* https://www.ti.com/lit/ug/sbou052a/sbou052a.pdf?ts=1620184069346&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTINA-TI

[7] Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling, https://www.iso.org/standard/63648.html

[8] Analog Discovery 2, User Manual, *Diligent*, https://reference.digilentinc.com/reference/test-and-measurement/analog-discovery-2/reference-manual

Appendix

A. *Design Alternatives*

There are many alternatives to smart home systems already on the market, but they are inadequate for widespread use. One subset of the market is occupied by proprietary solutions offered by companies such as Honeywell and Siemens, and aimed at automating large non-residential buildings. These systems are expensive, and because they are proprietary it means the building owner is locked dealing with a particular company. The other subset of the market is aimed at small home automation improvements in residential homes, examples being the Google Nest thermostat and Google Home. These devices are not comprehensive enough to control an entire building like the non-residential solutions are, are still expensive, and use wireless communication which can be at times unreliable. Given the cost and scale of some of these systems, it was unrealistic to attain them to evaluate them ourselves.

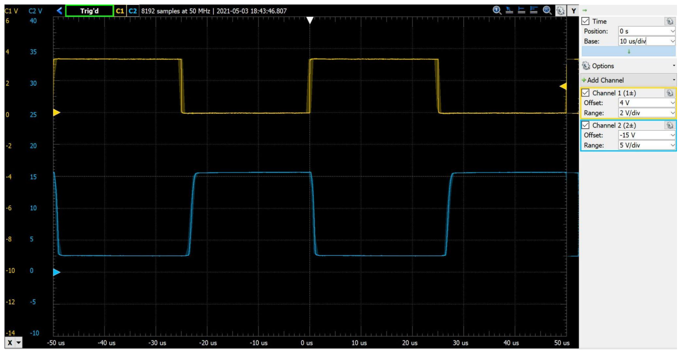
These existing solutions were inadequate given the goal of this project was to create a low cost, reliable, standard-interface smart home system.

B. *Technical Standards*

To communicate between nodes, this project uses the CANBus protocol. This is a two wire serial communication protocol, with common use cases in industrial and automotive applications. The CANBus protocol adheres to the underlying ISO 11898 series standards. These ISO documents set the physical layer and signalling standards for the CAN protocol [7].

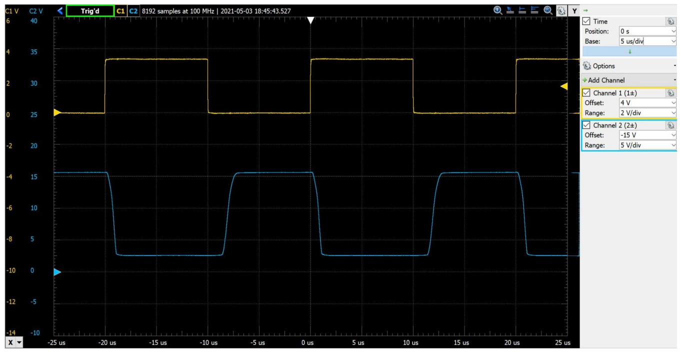
C. *Testing Methods*

Typical CAN bus systems can operate at 1Mbit/sec. Our system operates at a lower baud rate compared to other communication systems. To show how increase in frequencies affect the operation of our transceiver, we used a signal generator running at various values and measured the resulting effect on the PCB TX and Data+ pins. We started with our designed frequency of 20khz. The yellow line, ranging from 0 to 3.3 volts, shows the voltage of the PCB TX while the blue line, ranging from 0-16 volts shows voltage on the data+ pins. *Show in figure 18*, we have a clearly defined square wave. The following figures show what happens when frequency is increased. Around 250kHz, the rate is fast enough to prevent the data+ pin from reaching its 16 volt nominal value. This test was performed using a signal generator and oscilloscope on an Analog Discovery 2 [8]


Trig'd C1 C2 8192 samples at 50 MHz | 2021-05-03 18:43:46.807

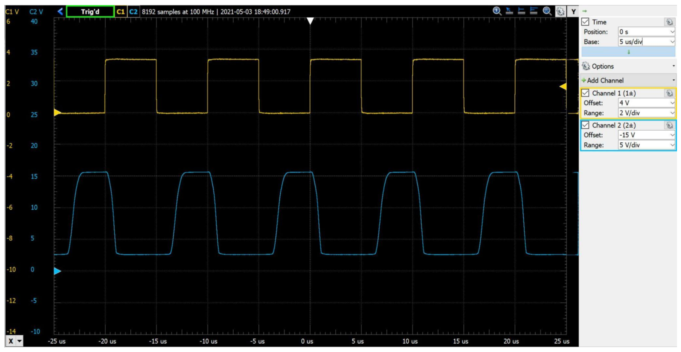
Time — Position: 0 s; Base: 10 us/div
Channel 1 (1±) — Offset: 4 V; Range: 2 V/div
Channel 2 (2±) — Offset: -15 V; Range: 5 V/div

Figure 18: 20kHz (Yellow TX pin, Blue 16 volt data+ pin)


Trig'd C1 C2 8192 samples at 100 MHz | 2021-05-03 18:45:43.527

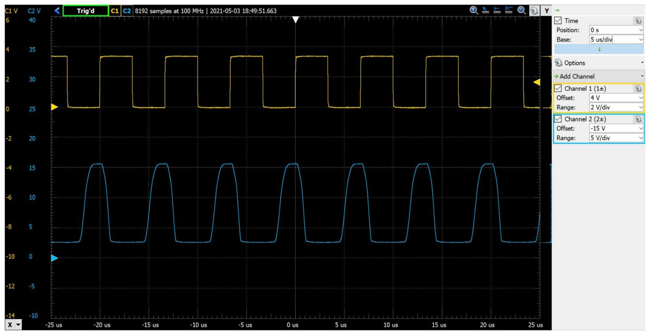
Time — Position: 0 s; Base: 5 us/div
Channel 1 (1±) — Offset: 4 V; Range: 2 V/div
Channel 2 (2±) — Offset: -15 V; Range: 5 V/div

Figure 19: 50kHz (Yellow TX pin, Blue 16 volt data+ pin)


Trig'd C1 C2 8192 samples at 100 MHz | 2021-05-03 18:49:00.917

Time — Position: 0 s; Base: 5 us/div
Channel 1 (1±) — Offset: 4 V; Range: 2 V/div
Channel 2 (2±) — Offset: -15 V; Range: 5 V/div

Figure 20: 100kHz (Yellow TX pin, Blue 16 volt data+ pin)


Trig'd C1 C2 8192 samples at 100 MHz | 2021-05-03 18:49:51.663

Time — Position: 0 s; Base: 5 us/div
Channel 1 (1±) — Offset: 4 V; Range: 2 V/div
Channel 2 (2±) — Offset: -15 V; Range: 5 V/div

Figure 21: 150kHz (Yellow TX pin, Blue 16 volt data+ pin)

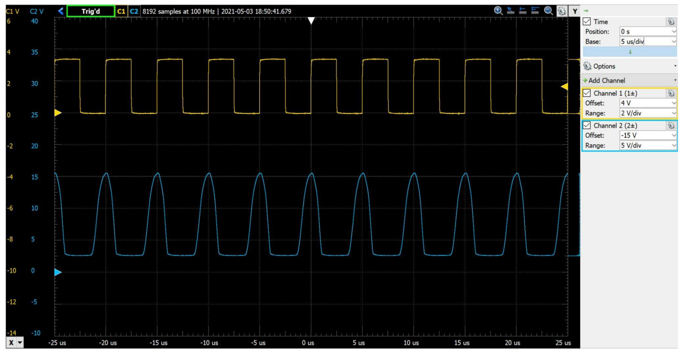


Figure 22: 200kHz (Yellow TX pin, Blue 16 volt data+ pin)

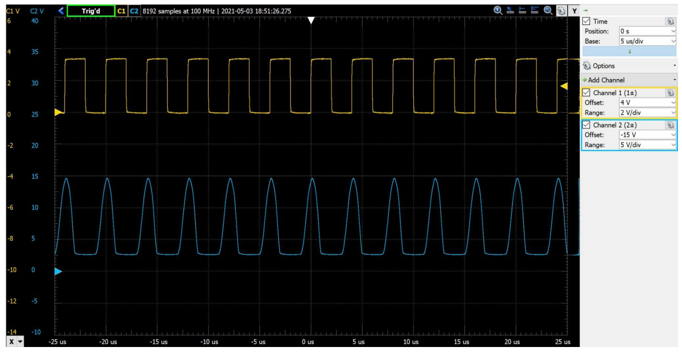


Figure 23: 250kHz (Yellow TX pin, Blue 16 volt data+ pin)

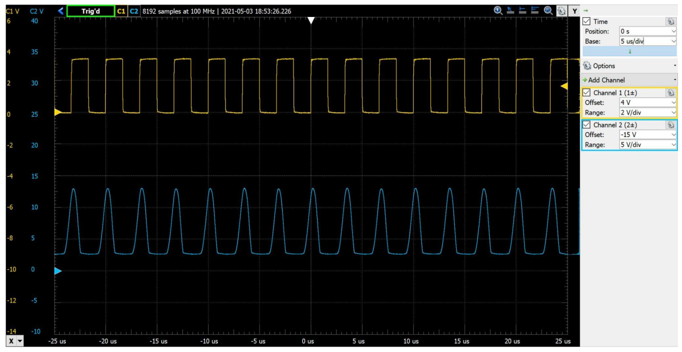


Figure 24: 300kHz (Yellow TX pin, Blue 16 volt data+ pin)

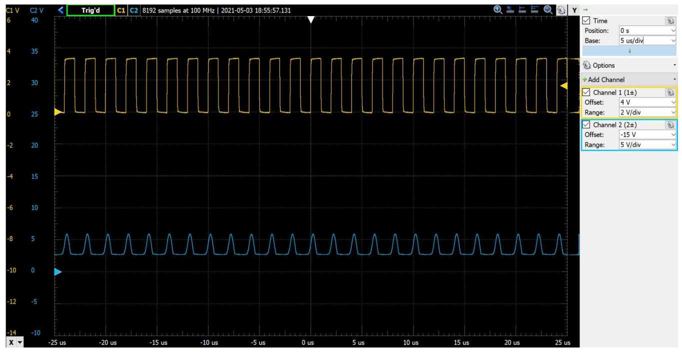


Figure 25: 500kHz (Yellow TX pin, Blue 16 volt data+ pin)

As the rate continues to increase, the peak voltage of the data+ pin will continue to drop. This drop in voltage will render the network inoperable. The purpose of this test shows what happens when frequency is too high. At lower frequencies, we are afforded more room for error. Since this test was performed using short jumper cables, this test did not account for longer lengths of wire. Longer runs of cables will have larger voltage drops. Having lower frequencies also allows us to successfully operate our system with a maximum node-to-node length of 1000 feet.

D. Project Expenditures

5 XMC4200 Dev. Board	$ 298.10
Transceiver components	$ 18.92
PCB Fabrication	$ 23.34
Step Motors	$ 12.50
Model Window components	$ 15.00

E. Project Management

Michael Chan, EE - Team Coordinator, Transceiver design
Dingbang Chen, CSE - Budget Lead, Output module
Nathan Johnson, CSE - Network Integration Lead
Tien-Li Shen, CSE - PCB Lead, Input module

In this SDP project, each member played an important role, and everyone was very united, and the final presentation showed a very satisfactory result for everyone. First of all, everyone is very active on the slack channel. Michael Chan was actively organizing and reminding everyone of each milestone that was approaching, and also sorting out the key points for each of our weekly meetings. As a person with the most C/C++ background in our group, Nathan Johnson actively explained coding techniques to the rest of our group members and came up with visions for the direction of the project. Tien-Li Shen also actively put forward suggestions and ideas throughout the development process. Our last excellent mock window was made by him, taking the initiative to spend the weekend. Dingbang Chen enthusiastically took on the role of building the team's website, where he managed and displayed all of the documents that our team generated. He also kept track of the team budget throughout the year. When determining the direction of the project in the early stage, everyone met on Zoom; when we needed to practice in the later stage, everyone actively gathered together in the SDP lab to efficiently complete the goals and tasks we set. There were points during this year where communication broke down and tasks were not completed to each other's expectations, but our team was able to come together and overcome these setbacks. Most of the issues came from mismatched expectations, and were solved through active communication on Slack and Zoom. Throughout the process of the whole project, everyone contributed their strengths as much as possible and was very united from beginning to end.

F. Beyond the Classroom

Michael Chan - My primary responsibility in this project was designing the physical layer that interfaced with CANbus. The beginning of this project was a steep learning curve. The tools necessary for me in this project included learning how to operate the TINA software for designing the circuit. I've learned how to solder many different components, SMT and through-hole on the PCB. Having multiple PCBs to solder gave me an opportunity to hone this skill with plenty of practice. Being remote for a majority of this project provided me with insight on how to communicate with team members via telecommunication and slack. Due to the pandemic, I see many technical companies switching over to permanent remote work.

Dingbang Chen - In this project, my main responsibility is to be responsible for the development of the stepper motor part. After comprehensively considering the advantages of various motors, we decided to use a stepper motor as the driving force of the window. The main reason is that the excellent power-off stop torque of the stepper motor can ensure maximum safety. In this project, we used the DAVE development IDE. Since we have never touched this development platform before, it becomes more difficult to be familiar with the IDE at the beginning. Later, because traditional stepper motors have a lot of excellent open source header files, but we have to code the operation sequence of the entire motor ourselves, which also allows me to learn a lot in it. Not only that, because of the help of EE teammates throughout the process, I also actually operated a lot of things that I had never done before.

Nathan Johnson - Overall, my central responsibility for this project was integrating code from Dingbang and Tien into a final, working system. This was challenging because it seemed like bugs that we hadn't seen before would appear once we started combining all our separate code. One part of the project I feel better at now is debugging distributed systems. Because we had many microcontrollers each doing their own thing, the global state machine for the system was fairly complex. Being able to understand a large system like this will definitely be useful in my future professional career. A resource I found helpful was the M5 staff. There were definitely times during the project that I appreciated being able to direct message Shira or another staff member on Slack to ask a question.

Tien Li Shen - My main responsibility was in the sensing systems and PCB design. For the PCB design, I had to spend a lot of time watching and rewatching the PCB tutorials made by the staff at M5. I found the office hours for Altium to be very helpful. The M5 staff were very forward and respectful in helping me with each step. SDP offered me a year long opportunity to grow in the areas of team working and public speaking.