UMassAmherst
The Commonwealth's Flagship Campus

SDP '21 - Midway Design Review - Active Windows

Team 6

Jonathan Clifford, Frank Cremonini, Griffin Manns, Connor Moore
Advisor: Professor Arman Pouraghily
Manhattan2 CTO & Contact: Mr. Glenn Weinreb

Overview

- Problem Statement
- Project Overview
- Our Solution
- Project Specifications
- Updated System Design
- MDR Accomplishments & Prototypes
- Hardware Plan For FPR
- Project Management
- Gantt Chart
- Project Expenditures

Problem Statement

"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."

Project Overview

- Window Operation
- We are focused on contributing to window automation in order to try and combat climate change.
- To do so, our team is focused on developing the end to end communications that this proposed system will utilize to communicate internally and externally.
- This is done via CAN internally and Amazon Web Services (AWS) externally, both to forward system status as well as User commands to control and monitor the window.

https://www.pellabranch.com/webres/Image/misc/2018-window-glass-option-header.jpg

Our Solution

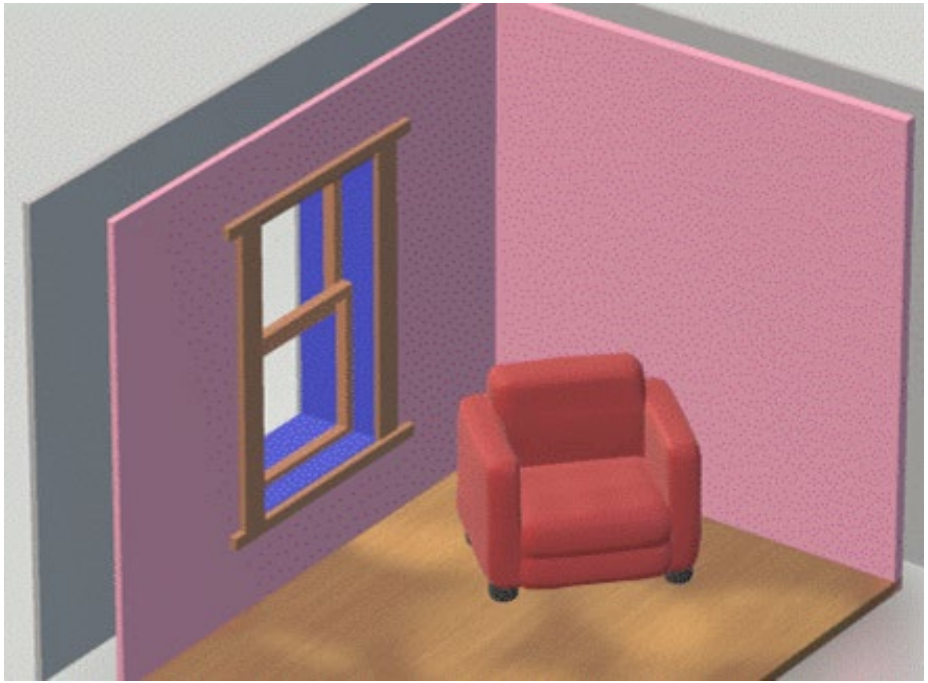
Windows are “Dumb”, we will attempt to make them more “Active” by allowing automated remote operation of the window.

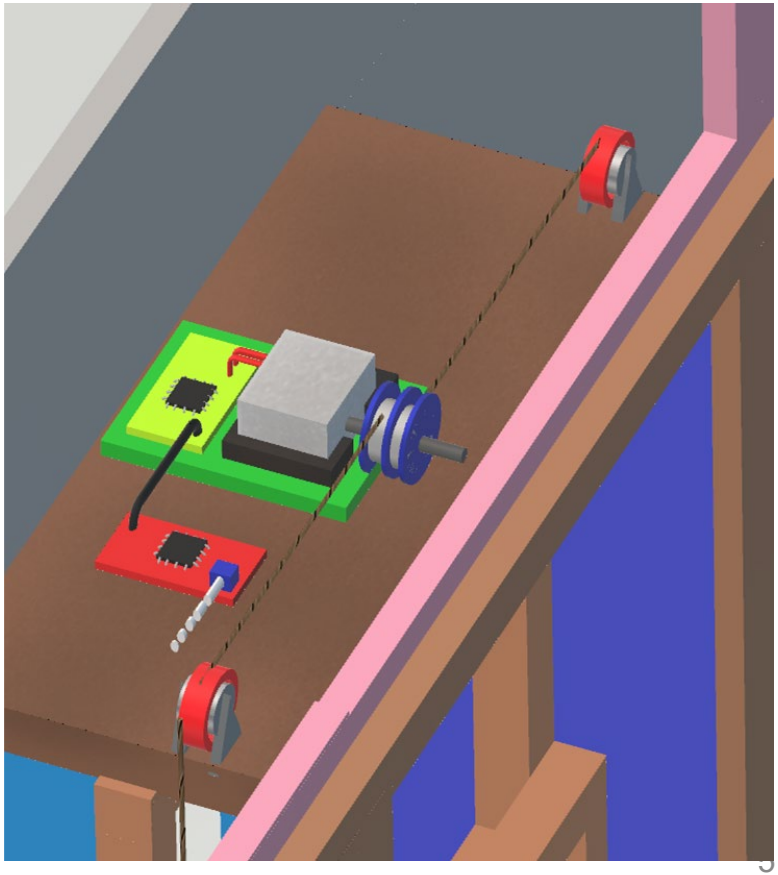
To achieve this we have the following components:

- **Android App** for the user to interface with the system and operate the window.
- **Cloud Connection** → Monitor and Control from anywhere.
- If Cloud is down, locally we have:
  - a manual override switch.
  - direct connection between the Android App & the system.
- **Network Master Controller** to control the local network and interface with Cloud services
- **Motor Module** to operate window.

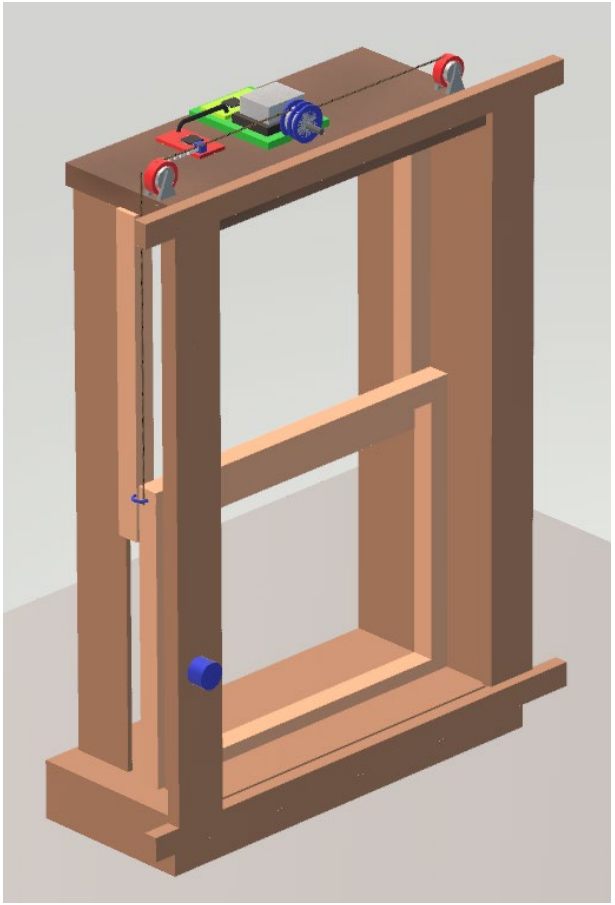
https://backtest-rookies.com/2018/10/26/tradingview-opening-a-window/

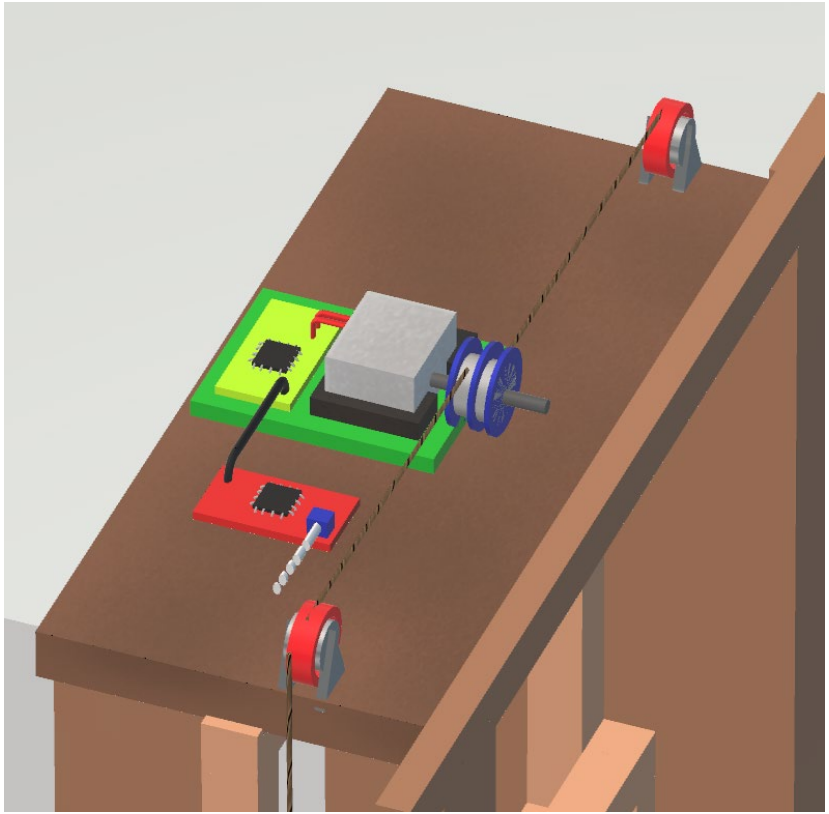
Our Solution in The Field





Model Window:



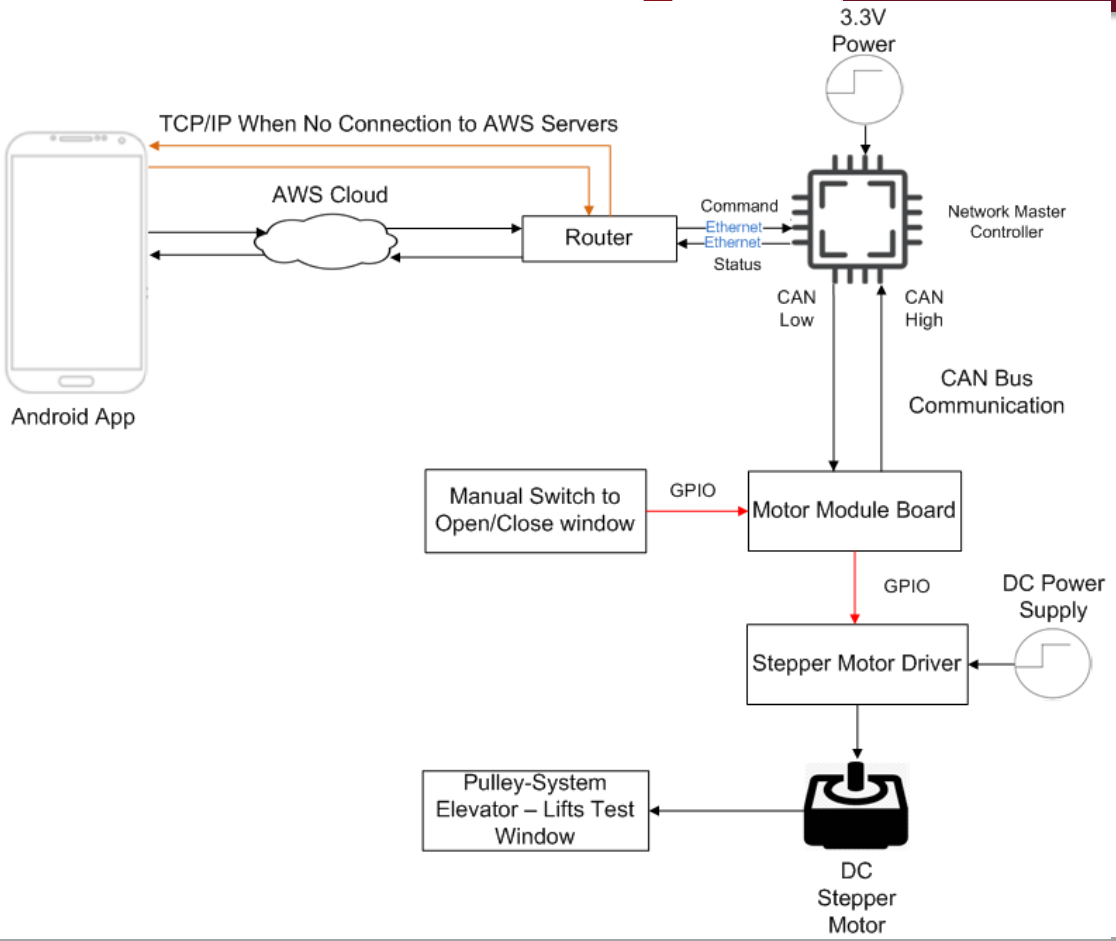


System Specifications

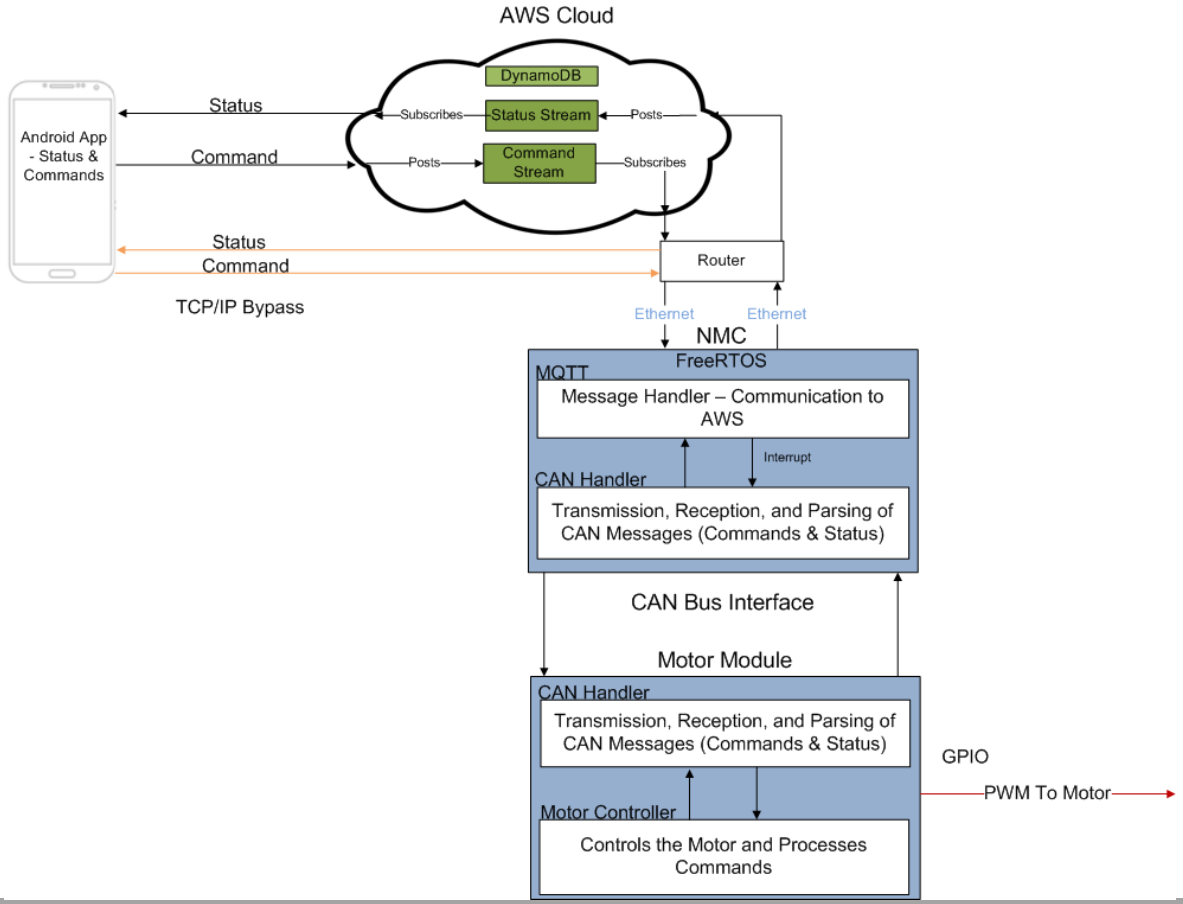
The Active Window Subsystems consisting of the Network Master Controller, IoT/Phone Application, and the Motor Module will meet or exceed the following specifications:

1) Capable of receiving signals from Amazon Web Services (AWS) for direct communication between the user and the rest of the system.
2) 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.
3) User interaction via an Android Phone Application that transmits data via AWS to the controllers. This Application will include a user interface to control and monitor window status.
4) The **Motor Module's** size is at least 14cm x 8cm in order to maximize installation in a home.
5) DC Stepper Motor and Driver, will utilize a 24V power supply and use a pulley system solution that can adjust the window via commands. This pulley system will physically lower and raise the test window once it receives the proper command.
6) In addition, there will also be 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.
7) 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.**

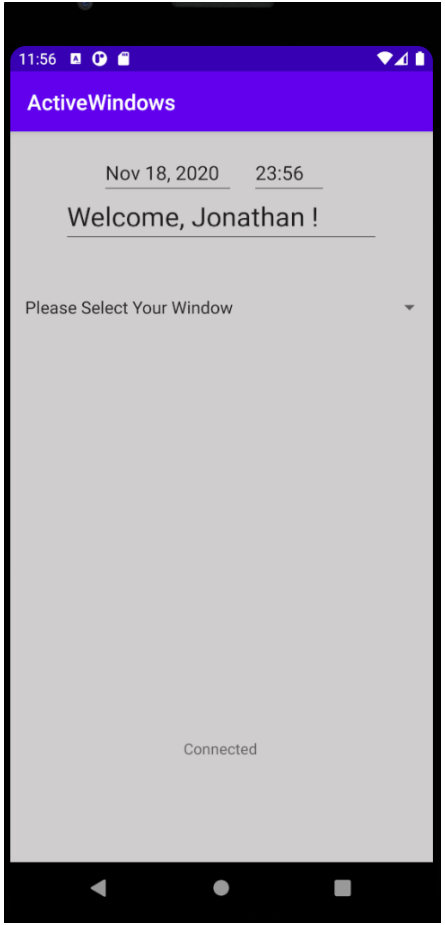
Updated Hardware Block Diagram

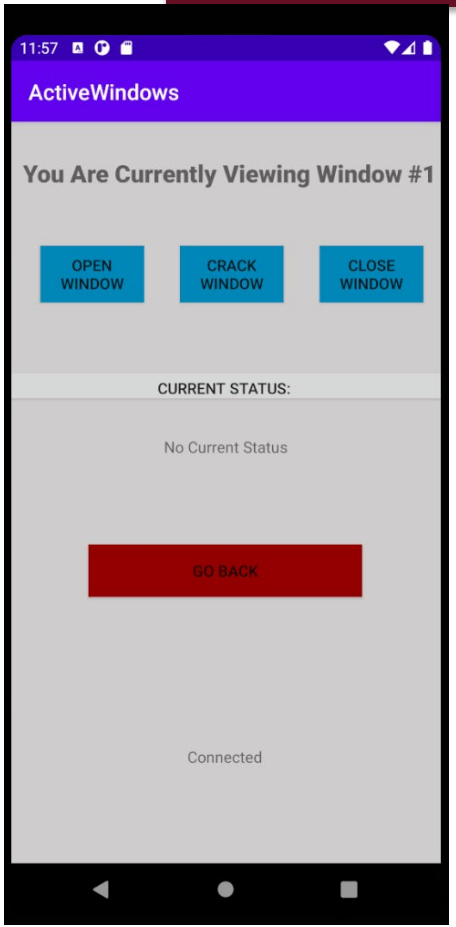


Updated Software Block Diagram



Updated Application Interface





MDR Deliverables

As promised, this is what we will be delivering for this MDR:

Deliverable Number	Deliverable	Engineer Responsible
1	A fully functioning connection between AWS and the Android Application.	Jonathan
2	The Network Master Controller is capable of connection to AWS and can transfer data.	Connor
3	The Network Master Controller is capable of direct connection and sending commands to the Motor Module via CAN Bus.	Frank
4	Motor Module can respond to a command sent from the Network Master Controller	Frank
5	**The Motor Module can drive and control the Stepper Motor**	Griff

MDR Demo - App & AWS

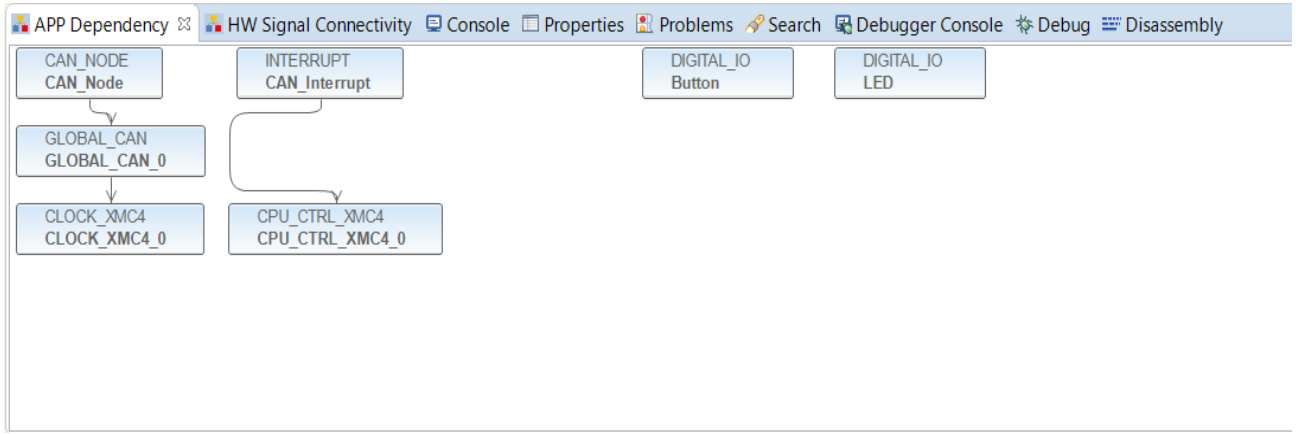
- Focused primarily on being able to establish communication between a simple Android App and AWS.
- Use of IoT Core & Amazon Cognito to properly communicate from the App to AWS.
- Amazon Cognito establishes “Roles” to communicate between the two - support for Authentication.
- App reads the most recent message from AWS, and upon user input sends a command for selected window.
- Issues: A noticeable delay & will only send a message after hitting the buttons once.

MDR Demo - NMC & AWS

- Focus was to establishing Connection between NMC and AWS Iot Core
- Simulated response from Motor Module command from AWS is printed to debug console and then published to windowStatusTopic
- Kept track of state variable on the NMC on the status of the LED
  - State table will be held in AWS in future production

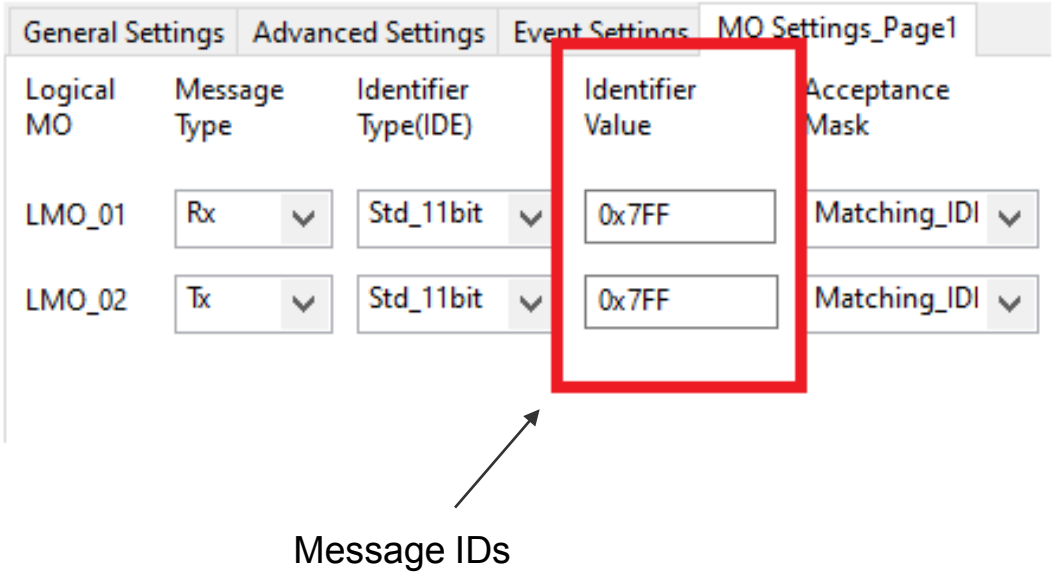
MDR Demo - CAN Bus Overview

- DAVE Graphical Programming
- Built on CAN Example, basic button functionality
  - Sent continuous messages based on button → Changed to toggle on button, acknowledge messages, print CAN message information

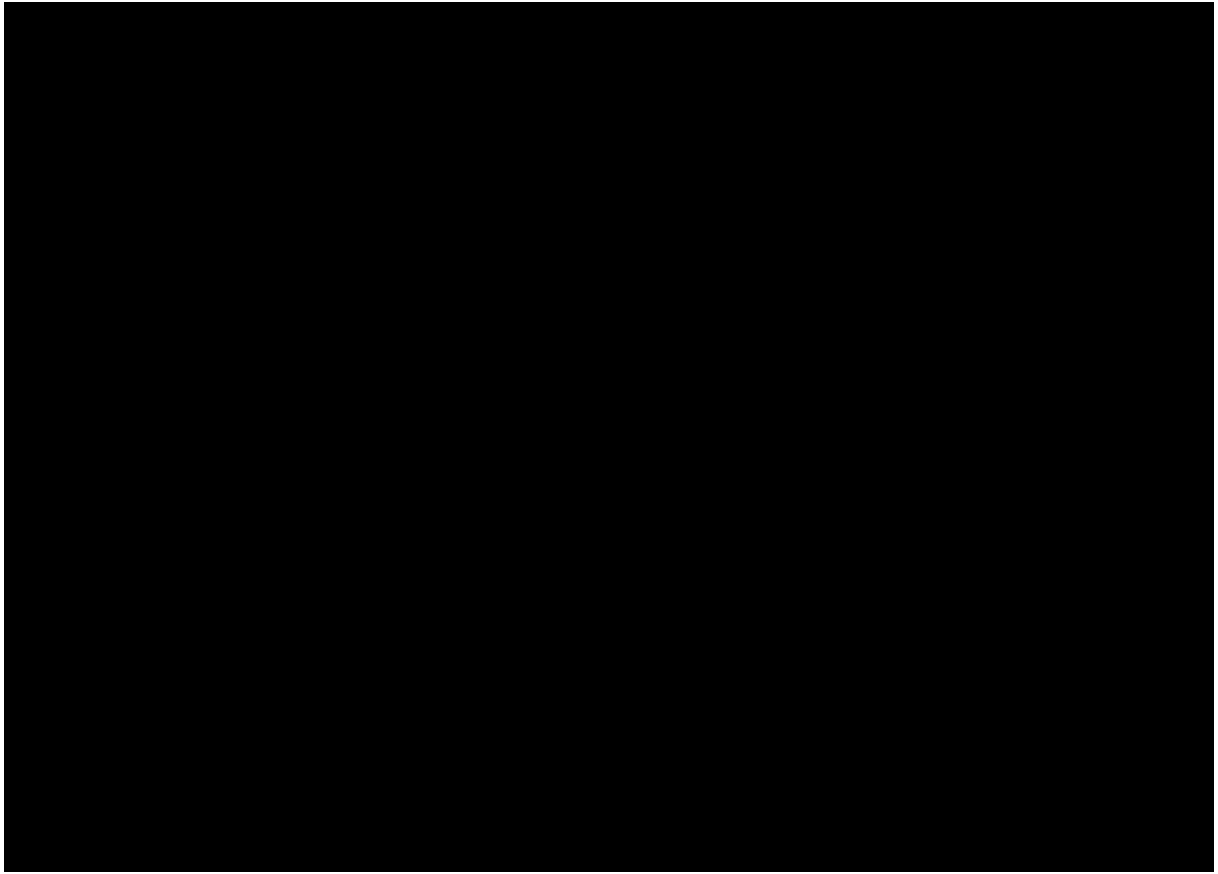


MDR Demo - CAN Bus Detail

- Receipt of CAN message triggers hardware interrupt, handled in code.



MDR Demo - CAN Bus



MDR Demo - CAN Bus Challenges

UMassAmherst

Challenges

- Wrong Cable
- Wrong Data Sheet
- Limited Support

Solutions

- Oscilloscope and Jumper Wires
- Trial and Error
- Lots of time and effort

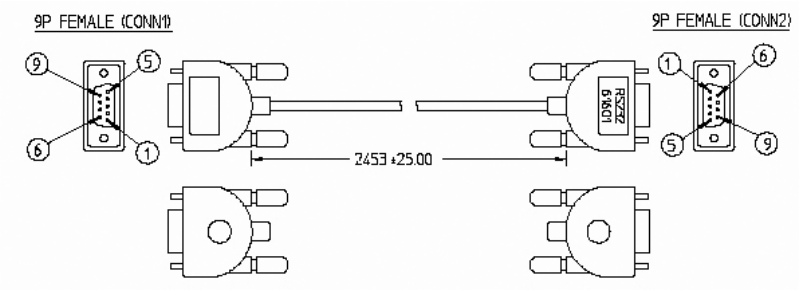
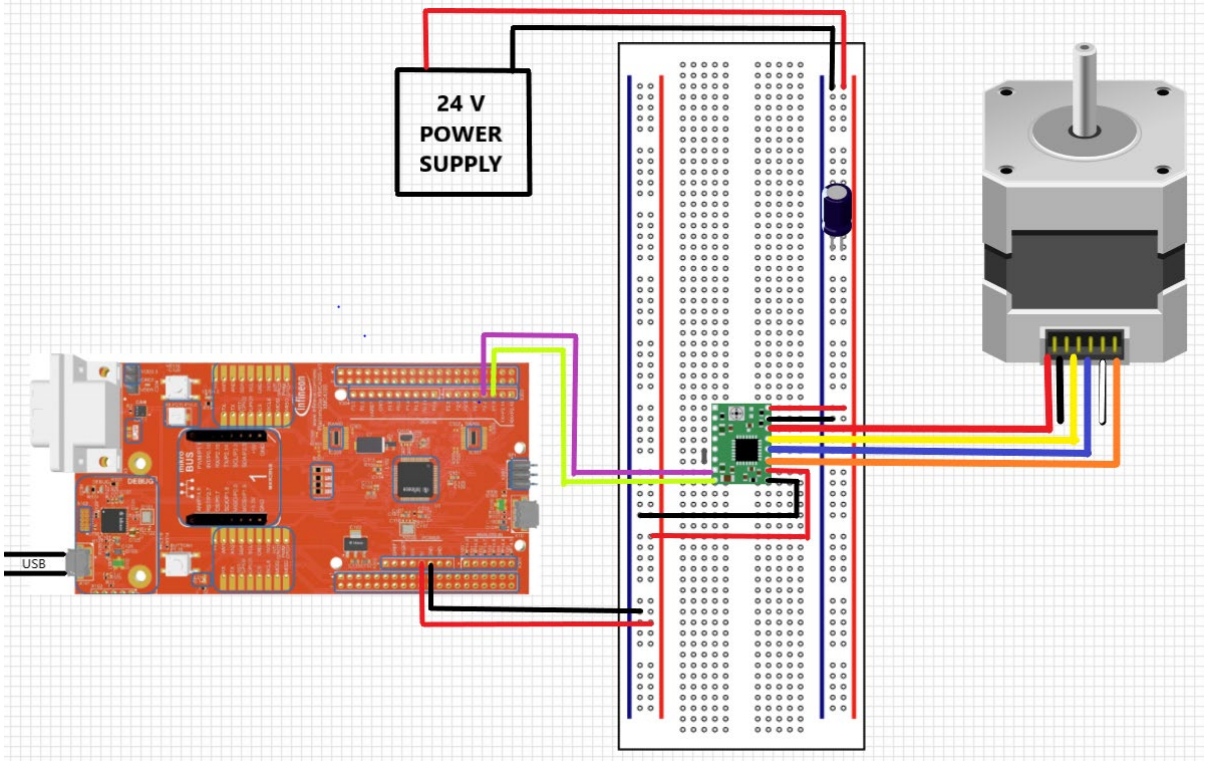


Figure 3 – Keysight RS232-61601 Cable Assembly

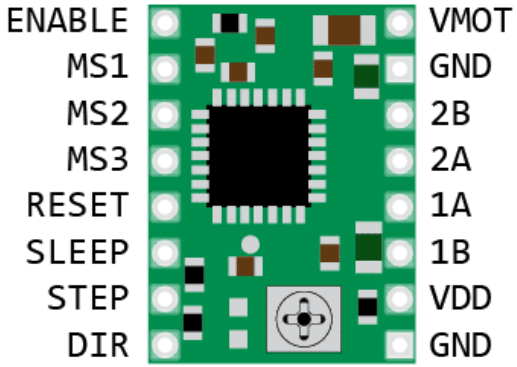
Keysight RS232-61601 Wire Connection Table				
Instrument Connection	CONN1	Wire Color	CONN2	PC Connection
DCD	1	Black	1	DCD
RX	2	Brown	3	RX
TX	3	Red	2	TX
DTR	4	Orange	6	DTR
GND	5	Yellow	5	GND
DSR	6	Green	4	DSR
RTS	7	Blue	8	RTS
CTS	8	Purple	7	CTS
RI	9	Gray	9	RI
	Shell	GROUND	Shell	

Figure 4 – Keysight RS232-61601 Cable Wiring Diagram.

MDR Demo - Motor Module



A4988 Driver

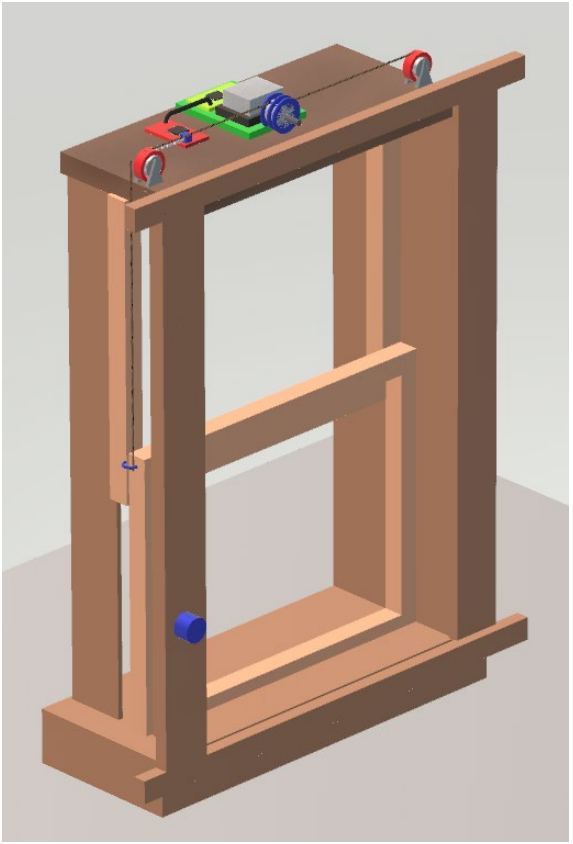


MDR Demo - Motor Module

- **Phase 1:** DIRECTION pin is set high to rotate the motor clockwise (opening the window)
- **Phase 2:** The PWM sent to the STEP pin is shut off to stop the motor
- **Phase 3:** The PWM sent to the STEP pin starts and the DIRECTION pin is set low to rotate the motor counterclockwise (closing the window)
- **Phase 4:** The PWM is stopped again, shutting off the motor

Hardware Plan For FPR

- Plan to create one PCB for the XMC 4200 (Motor Module)
  - CAN
  - GPIO
  - Debugger
- Plan to use XMC 4800 Evaluation Board
  - XMC 4800 PCB → Lots of overhead
  - Need approval from Course Coordinators



Hardware & Software Used

Software	Hardware
DAVE IDE Android Studio AWS MQTT Library Amazon Web Services (AWS) IoT Core (Message Streams) Cognito	XMC-4800 Microcontroller XMC-4200 Microcontroller A4988 Motor Driver 24V DC Stepper Motor TP-3005D-3 Power Supply CAN Bus Wires

Project Management

Jonathan Clifford
Team Coordinator & AWS,
Android Application Developer

Griffin Manns
Stepper Motor Module
& Altium Lead

Professor Arman Pouraghily
Advisor

Frank Cremonini
CAN Bus
Communication

Connor Moore
Network Master Controller
& Budget

Mr. Glenn Weinreb
Manhattan2 CTO & Contact

Gantt Chart - Spring 2021 - CDR

UMassAmherst

Task/Week Of	1-Feb	8-Feb	15-Feb	22-Feb	1-Mar	8-Mar
PCB Design & Ordering						
Manual Override Switch for Motor Module						
TCP/IP On NMC & Android App						
Final Revisements to CAN Protocol						
Full End-To-End Communication Established						
Free Week						
CDR Week						

Jonathan	Frank	Connor	Griff	All	Free	CDR/FPR

Gantt Chart - Spring 2021 - FPR

UMassAmherst

Task/Week Of	15-Mar	22-Mar	29-Mar	5-Apr	19-Apr
PCB - Installing Components & Debugging					
Window System - Design Updates if Needed					
Confirmation of End-To-End Communication Working on PCB					
Free Week					
FPR Week					

Jonathan	Frank	Connor	Griff	All	Free	CDR/FPR

Project Expenditures

UMassAmherst

Current Expenses:

Part	Cost ($)
XMC4800 Evaluation Board	$95.93
XMC4200 Evaluation Board	$57.85
24V 23KM-K066-00V Stepper Motor	$66.15
A4988 Stepper Driver	$7.45
Total	$227.38

Estimated Future Expenses:

Part	Cost ($)
XMC4800 Processor	$23.65 (per processor)
XMC4200 Processor	$6.31 (per processor)
Estimated PCB Costs	~$20.00
Pulley Parts (hardware, etc.)	~$30.00
CAN cable	$5.00
Model Window Parts (wood, hardware, etc.)	~$20.00 (sans glass)
Manual Override Switch(s)	$0.20 / Switch
Total	~$105.96

Overall Expenses (Current + Future)	~$336.34

Questions & Answers