Nozzle Flow Control for Agricultural Integration

ECE 499

6 August 2024

University of Victoria

Group Information

Group ID: 6

Faculty Supervisor: Dr. Mihai Sima - msima@ece.uvic.ca

Co-supervisor: Mike Wengryn - mike.wengryn@nozzleninja.com

Team Information:

S. No.	Name	V Number
	Jensen Gillett	V00931433
	Benjamin Lyne	V00914888
	Elliott Starchuk	V00929461

Acknowledgment

Thank you to Dr. Mihai Sima for assisting us as our faculty supervisor and providing feedback on our project.

Thank you to Mike Wengryn from Nozzle Ninja, for providing the project idea and existing components to use as reference for the design.

Thank you to Viktor Sarnev for providing us with 3D printed components on a very tight timeline.

Thank you to JLCPCB for manufacturing our PCB design.

Thank you to the UVic ECE Department and CEWIL for providing funding to cover the costs of this project.

Contents

Executive Summary

This ECE 499 project, Nozzle Flow Control for Agricultural Integration, focused on creating a low-cost system that can implement per-nozzle flow control by using existing Rate Control Units (RCUs) on tractors. Other solutions on the market are expensive, proprietary, and require replacing a tractor's original RCU for full functionality, thus a better solution was desired by Mike Wengryn of Nozzle Ninja.

For this project, a simple prototype system was designed, consisting of one 'control unit' that could connect to multiple 'actuator units'. Custom PCBs were designed using existing microcontrollers and logic components, then connected together using CAN bus. These were mounted into custom 3D printed enclosures for demonstration purposes. Software was written for the microcontrollers to allow communications between multiple units to turn nozzle flow on and off.

The project was an overall success with some limitations. Communication was successfully established between boards, allowing control of nozzles from the control unit to one actuator unit. However, issues in the communication between boards resulted in a late change to UART for communications, which would need to be resolved in a later hardware prototype. Overall, this prototype provides a good foundation from which to further refine and integrate a final product for the agricultural sector.

I Introduction

The Nozzle Flow Control for Agricultural Integration project aims to create a prototype of a low-cost, ISOBUS interface device that can replace expensive rate control units (RCUs) used in agricultural irrigation. Ideally, the device should be able to interface with off-the-shelf ISOBUS display units to individually control flow to nozzles on a tractor boom arm. Due to time and budget constraints, the prototype will focus on the individual nozzle flow control using its own command interface rather than an off-the-shelf ISOBUS display.

The project was suggested to the team by Mike Wengryn of Nozzle Ninja, an agricultural supply retailer in Stettler, Alberta. Mike sought to create a device that would interface between an ISOBUS display, something commonly found in tractors and other agricultural equipment, and the nozzles of a tow-behind sprayer. This would allow the use of Nozzle Ninja's spray recycling package, Re-Circulation, through existing equipment. Further development could be extended to other applications requiring precise sprayer control, reducing the cost of other types of per-nozzle control systems. Current per-nozzle control systems often require the replacement of the RCU with an inferior model, while burdening farmers with the high cost of the system. With a retail price of $12,000 [1], Nozzle Ninja is looking to reduce the cost to farmers while providing a superior system using integration to existing RCUs over ISOBUS, thus enabling farmers to make their operations more cost-effective and environmentally friendly [1].

All of the EGBC code of ethics applies to this project, as it is an engineering project in BC and therefore must adhere to the full code, covering competency, courtesy, professionalism, and regulations. However, the most relevant tenet is to "hold paramount the safety, health, and welfare of the public, including the protection of the environment" [2], since this project contributes to more efficient usage of pesticides and herbicides, therefore positively impacting public and environmental health. Additionally, lowering the cost of the system increases equity in the agricultural field, as it allows smaller farms better access to the technology.

Examples of potential buyers include:

- Current farmers who do not want to replace their existing RCU system, for tens of thousands of dollars, with a new system that may not integrate with their other existing equipment.
- New farmers who may not be able to afford a brand-new RCU control system, but still desire granular control over their spraying.

II Objectives

The high-level objective of this project is to provide a standalone system on top of an existing agricultural ISOBUS system, providing integration between existing rate-control units (RCUs) and commercial nozzle-body valves (actuators).

- Determine specifications of the project
  - The “what”: Individual nozzle control using ISOBUS commands
  - The “how”: hardware and software design
  - The cost
- Hardware implementation
  - Schematic and PCB design
  - Component selection
  - PCB finishing (component soldering)
  - Enclosure design and manufacturing
- Software implementation
  - ISOBUS communication between two microcontrollers using AgIsoStack++
  - Command unit logic, including input parsing and sending commands to the actuator unit
  - Actuator unit logic, including threading for stability and command processing
- Testing
  - Ensure all subsystems can communicate with each other
  - Ensure all outputs occur as expected

III Design Specifications

The project is to design a multi-unit system capable of controlling up to “60 nozzle bodies at 50 cm spacing on a 30 metre boom” [1], with a production cost of no more than $4000 [1]. A minimum working example for the prototype will be created, with extensibility in mind such that cost can be calculated. [1] also explicitly specified that ISOBUS (identical to CANbus from a hardware perspective) is to be used for future integration with RCUs and other in-cab devices over that protocol.

From [1], we further define that the system is intended to be deployed in an automotive, outdoor environment. The AECQ100 therefore provides a good baseline to design for, as compliance will lead to high reliability [3]. Since the devices are mounted to the boom and not near engines or heat sources other than the sun, they are in a relatively ambient thermal environment, so a requirement of Grade 2 (-40C to +105C) temperature rating is chosen.

A 12V input is available from the CANbus, and therefore the device must be powered off of 12V nominal, or 9-16V [3], unregulated source. Therefore, a 5v regulator is required to power digital components. However, the device will only perform basic digital operations, and thus requires little analog precision; regulated voltages therefore are permitted up to 10% ripple [4]. No sensors are used in this iteration of the product.

For extensibility, the system is real-time, but without hard deadlines, as Re-Circulation requires only that the nozzles be closed, but other systems require operation in motion. The slowest system is expected to be the physical actuator, with the fastest EH8 actuator taking 0.2 seconds per ¼ turn [5]. Therefore, a processing deadline of less than 100ms is sufficient, as it is an order of magnitude lower.

A summary of the requirements is as follows:

- Provide a standalone MWE, containing controller and actuators communicating over ISOBUS, extensible to up to 60 actuators
- Maximum production cost of $4000
- Temperature rating of -40C to +105C
- 5V regulated rail with 10% maximum ripple from an unregulated input in the range 9-16V
- Software deadline of 100ms

IV Literature Survey

Although a niche market, products fulfilling the goal of this project do exist; because of the small market, the figures can be magnitudinous. One such product is Pentair's Hypro system (Fig. 1). While pricing figures are not easy to locate, it is estimated that an entire system will cost between $25,000 and $50,000 (all prices in CAD unless otherwise specified) [1]. Individual components, such as an ISOBUS terminator, can cost upwards of $500 (433.46 USD [6]). Another option comes from Seletron and retails for $66,000 to $100,000 (48,000 to 72,000 USD [7]). The most expensive systems cost over $150,000 when factoring in advanced feature packages, while others increase overhead expenses due to using pay-for-use models [1].

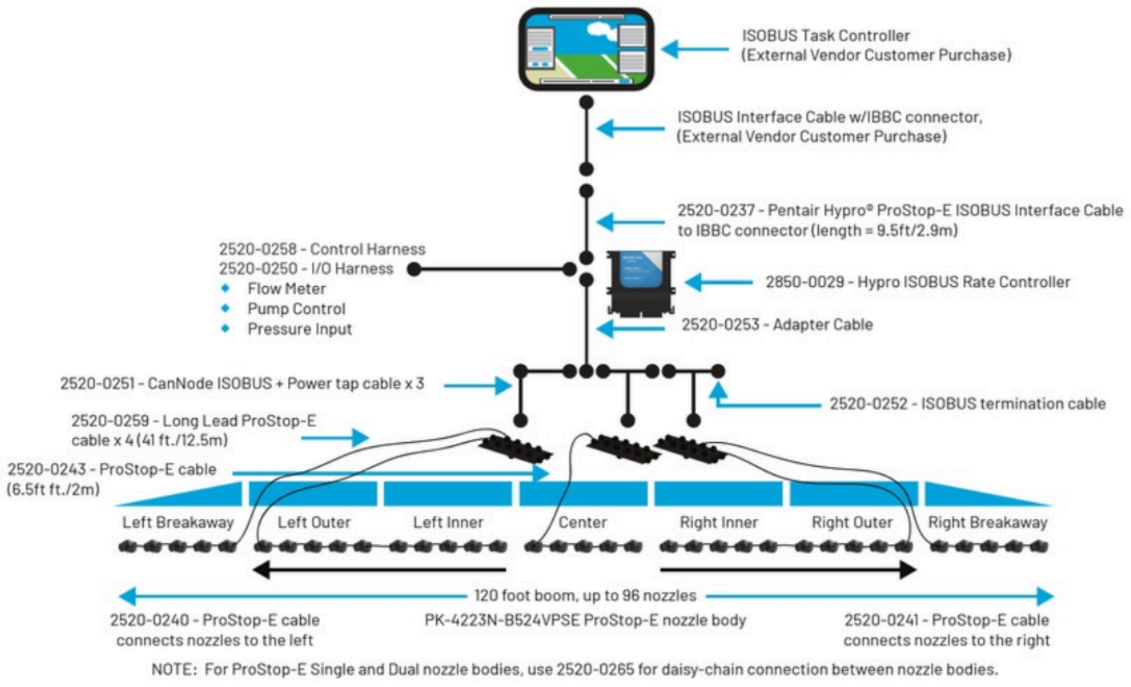

ISOBUS Task Controller
(External Vendor Customer Purchase)
ISOBUS Interface Cable w/IBBC connector,
(External Vendor Customer Purchase)
2520-0237 - Pentair Hypro® ProStop-E ISOBUS Interface Cable to IBBC connector (length = 9.5ft/2.9m)
2520-0258 - Control Harness
2520-0250 - I/O Harness
Flow Meter
Pump Control
Pressure Input
2850-0029 - Hypro ISOBUS Rate Controller
2520-0253 - Adapter Cable
2520-0251 - CanNode ISOBUS + Power tap cable x 3
2520-0252 - ISOBUS termination cable
2520-0259 - Long Lead ProStop-E cable x 4 (41 ft./12.5m)
2520-0243 - ProStop-E cable (6.5ft ft./2m)
Left Breakaway
Left Outer
Left Inner
Center
Right Inner
Right Outer
Right Breakaway
120 foot boom, up to 96 nozzles
2520-0240 - ProStop-E cable connects nozzles to the left
PK-4223N-B524VPSE ProStop-E nozzle body
2520-0241 - ProStop-E cable connects nozzles to the right
NOTE: For ProStop-E Single and Dual nozzle bodies, use 2520-0265 for daisy-chain connection between nozzle bodies.

Figure 1: Example nozzle body flow control system, using proprietary harnesses, cables, actuators, and RCU (adapted from [8])

Critically, none of these systems interface with existing RCUs, instead requiring farmers to purchase a brand new and proprietary replacement RCU. This subjects farmers to poor customer support, and unfamiliar or poorly designed UIs, thus impacting the usability of these products [1].

Therefore, Nozzle Ninja's approach addresses both the issues of cost and replacing the RCU, since the goal is to design with integration in mind, and provide a low-cost system.

V Team Duties & Project Planning

The duties of each team member are as follows:

Elliott has been assigned the software implementation portion of the project. The deliverables of the software implementation include:

- the implementation of ISOBUS communication using the AgIsoStack++ library
- software control of individual nozzle heads
- parsing input from a rotary encoder for nozzle selection
- display output programming
- ensuring all tasks are completed in a timely manner.

Ben has been assigned the hardware implementation portion of the project. The deliverables of the hardware implementation include:

- electrical component selection and procurement
- PCB design
- PCB finishing (component placement and soldering)
- physical ISOBUS implementation.

Jensen has been assigned the fill-in role of external communication and project planning (as team lead), and assistance with the hardware implementation. The additional deliverables of this include:

- communicating with UVic staff (assigned project TA Brian Zhu, project supervisor Dr. Mihai Sima, and course coordinator Sana Shuja)
- communicating with industry partner (Mike Wengryn of NozzleNinja)
- meeting and deadline scheduling
- document writing and review
- 3D modelling and printing of enclosures to house components
- electrical review (schematic and PCB)
- integration with industry equipment provided by NozzleNinja

No major challenges were faced in communication between team members, or between the team lead and either supervisors or the assigned TA. The team regularly communicated via the online chat platform Discord to update the status of their tasks and ask questions of other team members. All members executed on the tasks they were assigned. Challenges with the project implementation are discussed in the Design & Prototype section.

VI Design Methodology & Analysis

Hardware Specification

General Overview

The basis of the design is not dissimilar to existing designs (Fig. 1), though our design cuts out many unnecessary proprietary components. The most notable difference will be the integration with customers' existing RCUs, rather than requiring a proprietary controller. The proposed system will thus start at what Fig. 1 refers to as "CanNodes," providing a direct interface between the existing display or RCU to any 12V-controlled nozzle bodies. Since the nozzle-body "actuators" are not CANbus capable, "actuator units," will be used to interface the actuators to a central "command unit" that translates input from the ISOBUS display or RCU and transmits them to the actuator units over the CANbus (Fig. 2). For the initial prototype, the command unit will utilize a simple mechanical control system (e.g. buttons and switches), expecting more sophisticated integration in a future design.

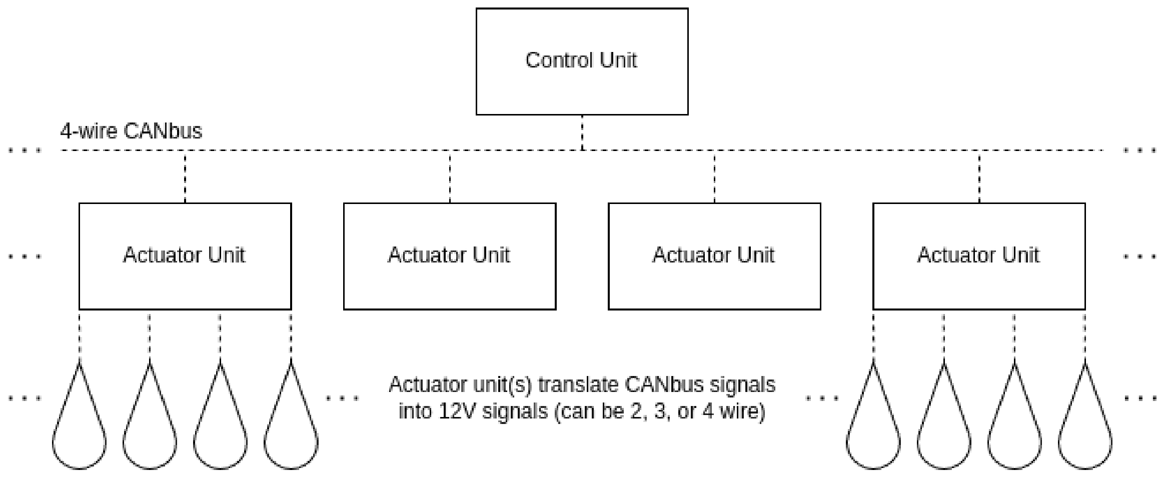

Control Unit
4-wire CANbus
Actuator Unit
Actuator Unit
Actuator Unit
Actuator Unit
Actuator unit(s) translate CANbus signals into 12V signals (can be 2, 3, or 4 wire)

Figure 2: Block diagram of proposed prototype system

Applying to all hardware, the automotive AECQ100 qualification is required, or qualified alternatives are planned, as modules are expected to be deployed in harsh outdoor environments. Additionally, all components must be available for future manufacturing, i.e. not marked as obsolete or unfit for new products. Cost is also considered, where other requirements do not supersede. Non-compliant components are selected due to availability from past projects, to be used for prototyping purposes only.

Component Specification

The overall components specification is made according to the following design (Fig. 3), which encapsulates the entire single hardware design (recall that the "role" is to be defined in firmware). Additional limitations and components are introduced as needed.

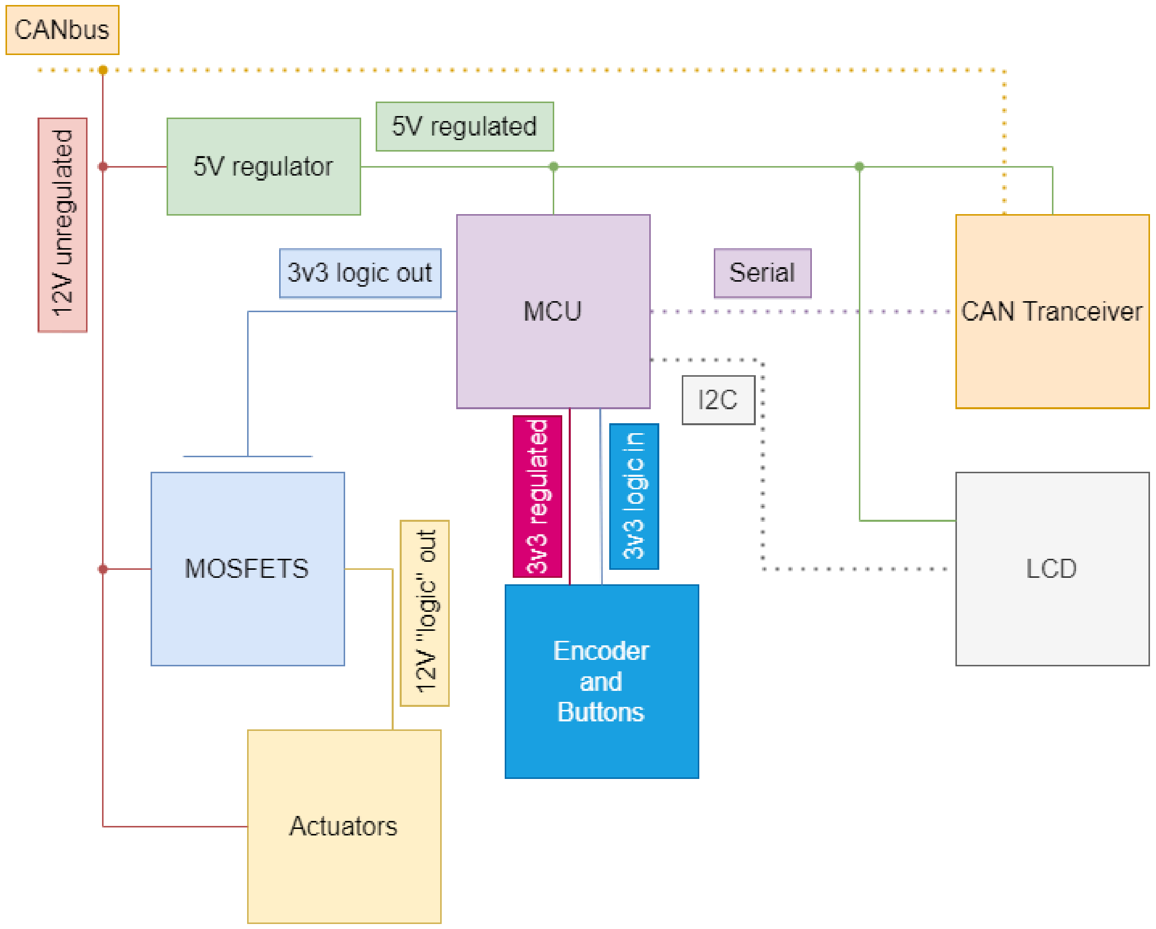


Figure 3: Block diagram of required components for prototype hardware

For each node (whether control or actuator) in the system, a microcontroller, or main control unit (MCU) is required. The Xiao ESP32-C3 is selected as the MCU for the prototype due to it being in possession; it is the only part in this prototype that does not meet the AECQ100 qualification, and is only rated for Grade 0 (-40°C to 85°C), meaning it would be replaced in a revised design with a Grade 1 temperature rated variant. This MCU adds a restriction that no more than 11 GPIO may be used. Additionally, due to minimum order quantity requirements of JLCPCB, it was decided to use a single PCB design and populate them according to purpose (command unit or actuator unit), thus restricting IO across all variants to those 11 GPIO pins (i.e. even if a component is unpopulated, we maintain that the GPIO still cannot be used for any other function). Under this unified design, the role of each unit is defined by the MCUs' firmware.

Finally, this MCU adds requirements that all logic signals must operate on 3v3, as they step down their input power to 3v3 logic.

The I/O requirements were provided by [1], who requested that the initial prototype use a standalone controller. Therefore, control and display mechanisms are required. To minimize pin usage, it was decided that a 2004 (20 x 4 line) LCD with an I2C interface and a rotary encoder with a built-in switch would be used for control and display purposes, as these used the fewest pins while also providing a minimalistic user interface. Furthermore, since the rotary encoder control is primarily implemented in firmware, extension to spot-spraying or integration with existing CANbus RCUs could be done in firmware. These components were selected purely as the lowest-cost versions available from Amazon or DigiKey, and are generic enough to add no additional requirements.

The ISOBUS requirement from [1] is also the basis for needing a CANbus transceiver (since the physical implementation of ISOBUS is CANbus). To handle differential 12 or 24V CAN signals with an MCU requires additional hardware to split the signals into proper RX and TX lines and to step down the voltages to a reasonable signal level. An ATA6561-GAQW-N was selected due to its low cost and adherence to other requirements. This device also receives power from the 5V rail to operate, and communicates with the MCU at 3.3V [9].

The largest current consumer on the 5V rail is the MCU, which draws a maximum of 200 mA [10]. The CAN transceiver has a maximum draw of 70mA [9]. Other devices on the 5V rail are expected to draw less than the MCU, as they are FET-based semiconductors. Therefore, a supply current of 1A is sufficient to guarantee a 2X amperage safety factor. A 10% ripple requirement is also imposed [4]. No sensors are in the design, but FETs will be controlled by the MCU's onboard 3v3 LDO, which is subject to the same 10% ripple requirement. A 12V (nominal) input is expected from the CANbus, and therefore a buck regulator was required due to the large step-down. Therefore, the 2A adjustable output voltage AP62200WU-7 buck regulator was selected, primarily due to cost factors. With a 20V absolute maximum DC input [11], there is a safety factor of 1.67X, which is acceptable since the 12V rail is from a vehicle battery, and therefore will trend lower, not higher, than the nominal voltage specification (and 20V is still within the specification by [5]). Per [11], the ripple is expected to be within 20mV under all expected operating conditions, thus satisfying the 10% (500mV) ripple requirement. The XC6210B332MR-G used on the Xiao board [12] has line regulation of 0.2%/V and 60dB of ripple rejection (at 1kHz) [13] and therefore is also expected to meet the 10% requirement.

The valve specs provided by [1] require a 12V control signal to actuate [5]. Therefore, 3.3V logic transistors are required to control signals tied to a 12V rail. A threshold voltage near 1V (30% of V_{DD} [14]), and maximum drain to source breakdown voltage, BV_{DS}, above 12V are required. The DMN2004WK-7 N-channel MOSFET was the lowest-cost solution meeting these requirements, with a 1.67X safety factor on BV_{DS} [15]. The low safety factor is acceptable here for the same reason as on the regulator.

Interrupts are to be used on the encoder input, and therefore hardware debouncing is required. Per [16] (Fig. 4), this means an inverter with a Schmitt Trigger is required. Since there are three signals to debounce (A, B, and the switch), it must have at least three inputs. The 6-channel CD74HC14M96 was found to be lower cost than 3-channel alternatives, while meeting the other requirements. The formulas used to calculate R1 and R2 are found in [16], and based on C = 22uF, calculated using an interactive spreadsheet.

Due to being in possession already, diodes were selected arbitrarily to be either 1N5817 or B5817W Schottky diodes, variants of the same device in different packages. However, to meet the AECQ100 requirement, PMEG2010AEJ,115 diodes will instead be used for the final design.

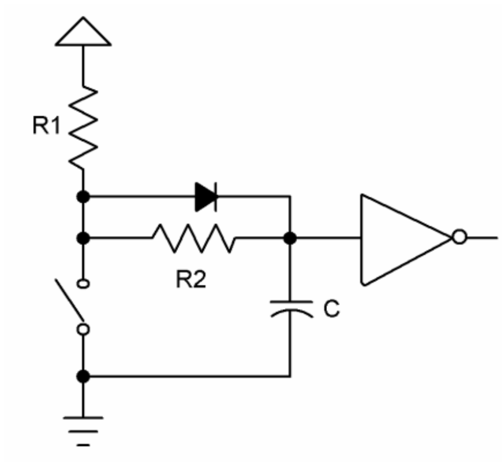


Figure 4: Debounce circuit (adapted from [16])

Finally, passive component selection was required. The exact values, where not addressed until now, are available from the respective datasheets, as the recommended application circuits were used.

For resistors, 1% 1/4W 1206 (3216 metric) was selected, since SMT is lower cost and: (a) 1% is a common tolerance value for SMT components, (b) the highest voltage is 12V and smallest resistor 3k, so 48mW is the max expected power through a resistor, therefore 250mW provides a 5X safety factor, and (c) a fairly large readily available SMT package was selected to aid hand soldering.

For capacitors, selection was less general due to costs. However, all capacitors are selected to be better than 10% 16V 0805 (2012 metric), for similar reasons to the resistors and since these encompass many common capacitors. Explicitly, the input capacitor on the 5V regulator, C1, has a rating of 25V to meet a 2X safety factor on the 12V rail.

Likewise, a 20% 5.8A nonstandard package inductor was selected due to cost.

Per [1], the primary form of connectivity is to be Deutch connectors. Based on the control schema outlined in [5], 2-4 line control systems are available, so a 4-position Deutsch DT series connector is selected, and will be panel mounted in final designs. Interfaces from the PCB to these panel-mounted connectors are required, so for simplicity four PTH are needed for each connector, though quick-connect tabs can be used in a future iteration to improve manufacturability.

Manufacturing Specification

The PCB is specified primarily in terms of trace widths, with 1 oz. copper thickness, as calculated using [17]. [5] states that an inrush current of ~10A is expected, so for safety the 12V rail is designed to deliver 10A at 20C rise, therefore the trace width from the CANbus to the nozzle connectors must be 5mm. Based on the maximum output, the 5V rail and input to the regulator from the 12V rail is designed for 2A at 10°C rise, and therefore is 0.8mm. Since the maximum single current draw is 200mA, all other traces are designed for an overkill 1A at 10°C rise, as board space is not a limitation, and are therefore 0.3mm. Likewise, since no high current vias are in the design, the KiCAD default netclass size is used.

Furthermore, for this prototype, manufacturing is done by JLCPCB. Therefore, strict adherence to the tolerances in [18] are followed.

Finally, the enclosure is specified as follows: M3 screws and standoffs are selected due to low cost and high availability on Digikey. The enclosure is to be 3D printed, and fit the final dimensions of the PCB, per the dimensions listed in the KiCAD PCB file itself. Likewise, the 5 M3 mounting points are to be dimensioned; standoffs are to be glued into a boss at these locations. The overall height must fit the combined height of the PCB and the LCD, to be dimensioned once the PCBs arrive.

Software Specification

General Overview

A main objective of the project is to create a device that can communicate over the ISOBUS protocol. Although programming tools do exist for implementing ISOBUS, most of them are proprietary and thus inaccessible to the team. However, an open-source library implementing ISOBUS called AgIsoStack++ was found and selected for use in the project [19]. AgIsoStack++ implements ISOBUS as a C++ library and supports a wide variety of systems, including ESP32 [20], making it a good fit for the project.

As AgIsoStack++ is implemented as a C++ library, the project's software must be implemented in C++. The prototype should use C++'s feature of C style linkage to try ensure compatibility with libraries written in C. C++'s backwards compatibility with C code will be useful, as it should allow the use of C functions provided by the MCU manufacturers.

Command Unit Software

The command unit will be the unit that a user interacts with in the finished prototype, so the software specification was designed with that in mind.

The command unit software is responsible for gathering two input types: a rotary encoder input, which is used to select the nozzle a user wants to interact with, and a button input, which is used to interact with the currently selected nozzle. There are two methods for gathering inputs, polling and interrupts. As the project aims to give real-time control of nozzles, interrupt-based input handling is used.

The command unit is also responsible for relaying information to the user via a small LCD display in the prototype. The display shows which nozzle is currently selected, as well as the nozzle's current status, and should update in real-time based on user input. The LCD will be communicated with via the I^2C bus. The I^2C bus was chosen due to the limited number of GPIO pins on the Xiao ESP32-C3.

The command unit should send commands via ISObus to the actuator unit, although only one command at a time. This limitation was chosen to potentially simplify implementation, as well as acknowledging that it is extremely unlikely a user will be able to input another command before the prior command has been sent over.

Actuator Unit Software

The actuator unit's responsibility is to receive commands from the command unit and act on those commands in a timely manner to control the nozzles. Ideally, the actuator unit makes use of two threads: a listener thread and an execution thread. The listener thread is responsible for listening for and receiving commands from the command unit. Each command received should be placed in a first-in-first-out (FIFO) queue. The execution thread would be responsible for executing the commands within the queue. By having two separate threads, the actuator unit should be able to continually receive commands even if the execution thread is working through a command queue. For stability, it may be required to implement two FIFO queues: one for the listener thread to load commands into, and one for the execution thread to read from. A two queue implementation may help avoid a situation where one thread modifies memory another thread may not have yet finished reading from, which could cause incorrect or invalid commands.

VII Design & Prototype

Completed prototype

In the completion of the prototype, there were some issues encountered that required input from all team members to solve. One major issue was the failure of a CAN bus transceiver without sufficient time to acquire a replacement. The transceiver failure meant that communication between the two planned units would have to be modified in order to have a working prototype. The chosen MCUs had the ability to communicate over UART (universal asynchronous receiver/transmitter), so a pivot was made to implement UART communication for the prototype. The team was able to reuse the MCU pins the CAN bus modules were connected to as UART pins, which meant minimal PCB modifications had to be made. Implementing UART in software for the chosen MCUs was well-documented, making implementation simple.

Another major issue faced was the behaviour of the rotary encoder on the command unit. The encoder was exhibiting unexpected behaviour, seemingly skipping steps and ignoring inputs. Using a desktop scope, it was discovered that the acquired rotary encoders did not behave in an expected manner, breaking the library that was previously chosen to interface with the encoder. With insufficient time to source a replacement encoder, there were multiple attempts to fix the encoder both in hardware and software, but to no avail. Eventually, the choice was made to rewrite the command unit firmware in a different framework to give access to alternative libraries. The firmware rewrite was successful, with there being a library available that allowed the team to mitigate the unexpected encoder behaviour in time for the presentation.

Hardware Prototype

Electrical Prototype

The hardware prototype consists of two boards, populated as necessary to form one Control Unit and one Actuator Unit, per the component selections in section VI. A carrier PCB is designed per Fig. 5, as shown in Fig. 6, 7.

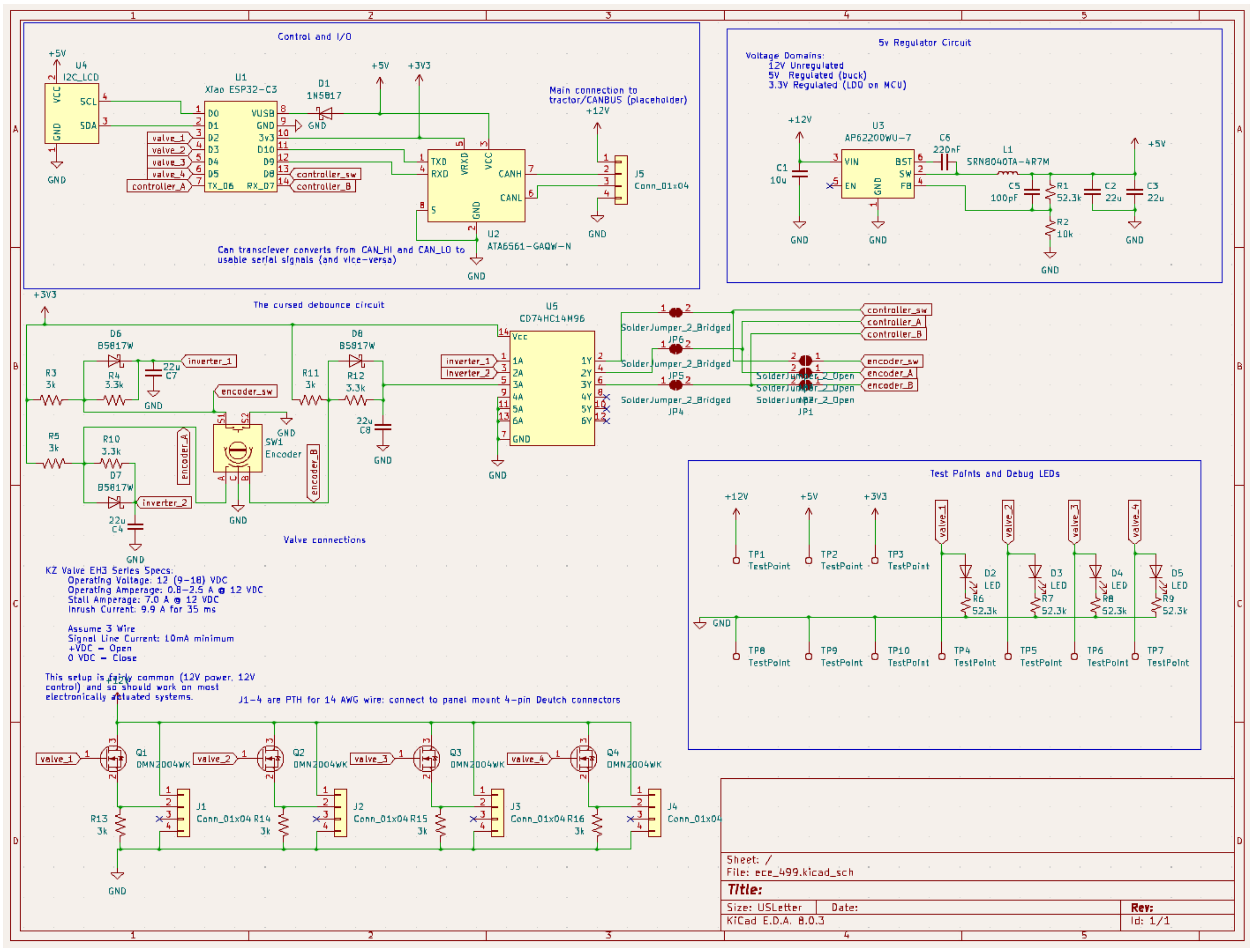


Figure 5: PCB design schematic.

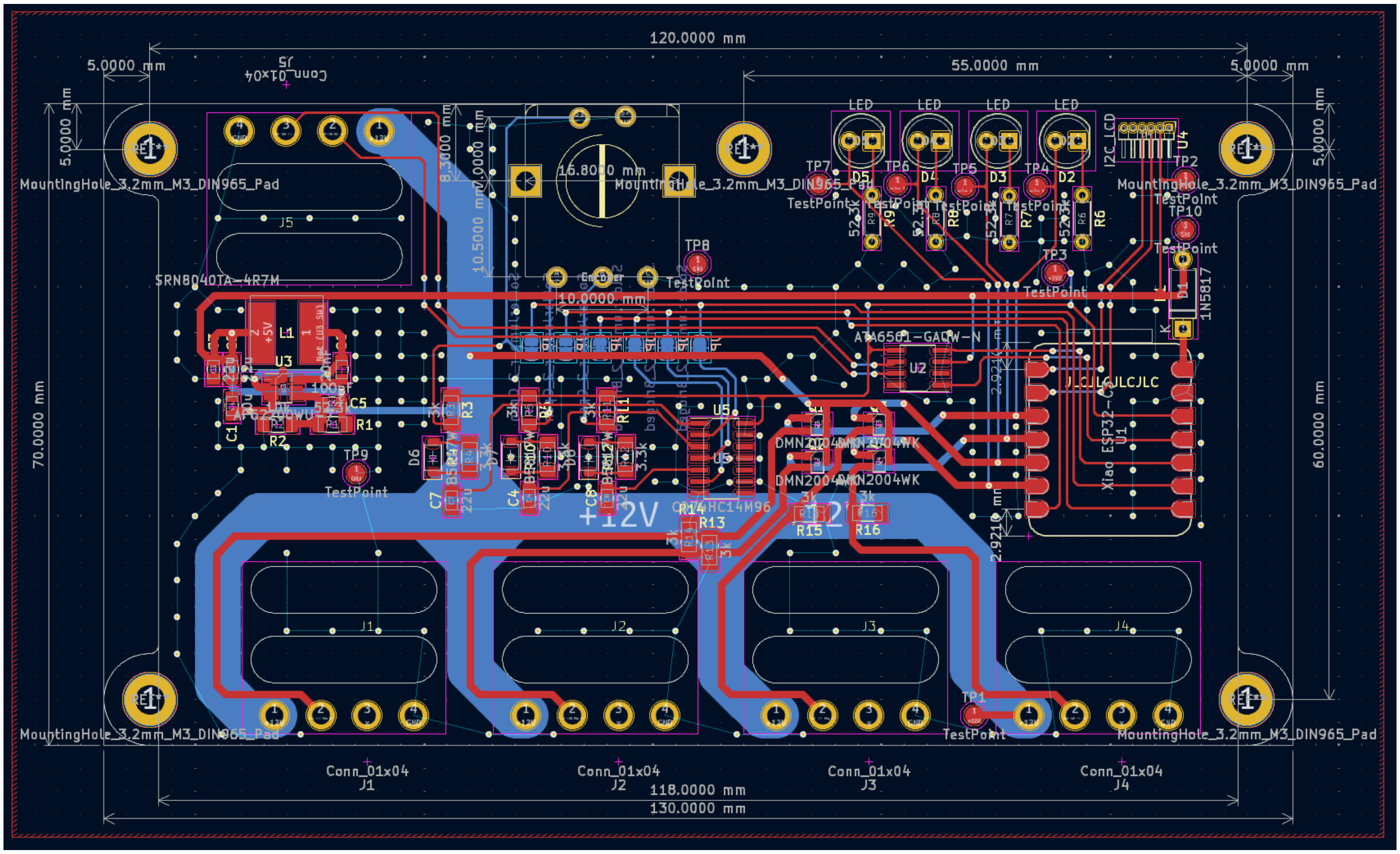


Figure 6: Prototype PCB layout.

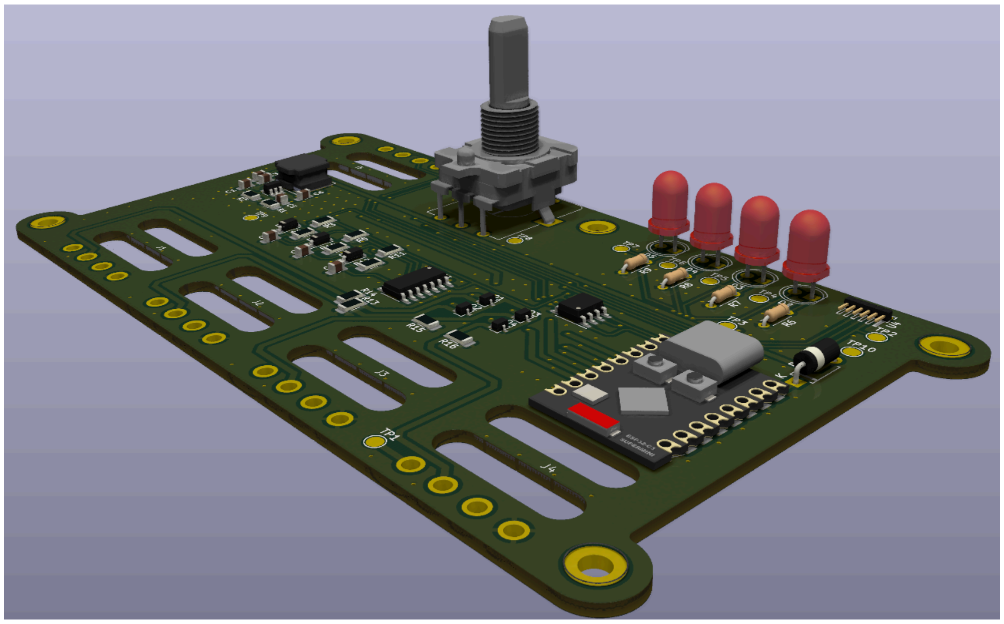


Figure 7: 3D model of prototype before fabrication.

There are a number of differences between the prototype and final design, some reflected in Fig. 5 and others due to post-fabrication related issues.

The most obvious of these are the LEDs and cable strain relief channels, most visible in Fig. 7. The indicator LEDs are in place of actual actuators, and are used on both the Command and Actuator units to indicate when the MCU has selected a specific nozzle to be opened. The Command unit has an LCD, on which it repeats this information using text, however, the Actuator lacks this LCD and therefore needed a way to show that the hardware had correctly selected a valve; LEDs on the GPIO lines controlling each nozzle was the simplest way of doing so. The stress relief channels at the edges of the board were included as the Deutch connectors used are panel mounted, and therefore must be wired into the board in a mechanically sound manner. Weaving a cable through these channels is the simplest way to provide stress relief (visible in Fig. 8, bottom right), thus ensuring the cable-to-board connection is mechanically secure; a more manufacturable solution would be to use quick connect spades, however, this would have increased the BOM and made manufacturing the prototype slower.

Fig. 9 also shows some of the post-manufacturing changes required. The incorrect LCD header footprint was included in the final manufacturing files sent to JLCPCB, so to work around, the correct sized header was soldered to an unused nozzle port, and jumper wires connected to their respective pins. Following the destruction of multiple boards and CAN transceivers during smoke testing of the Actuator Unit, more jumpers were required, bridging the TXD and RXD nets directly to the CANH and CANL pins, so that a UART communication could be retrofit, thus eliminating the need for the (now destroyed) CAN transceivers (Fig. 8). This means the ISOBUS requirement is being postponed for a future hardware iteration.

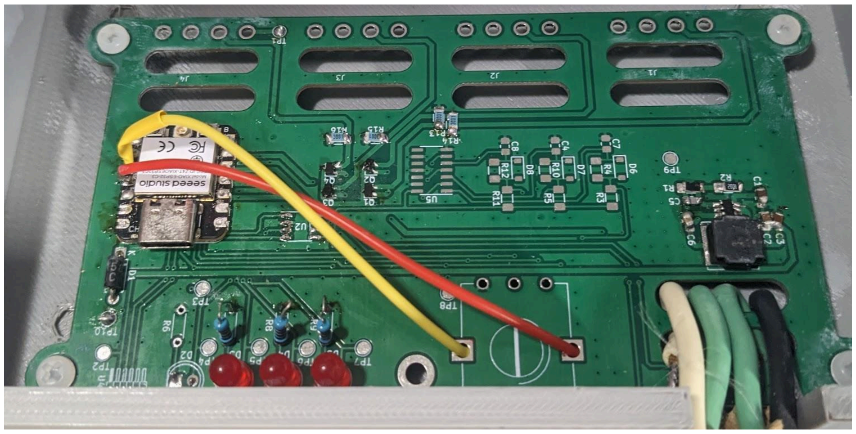


Figure 8: Modified assembly of Actuator Unit PCB

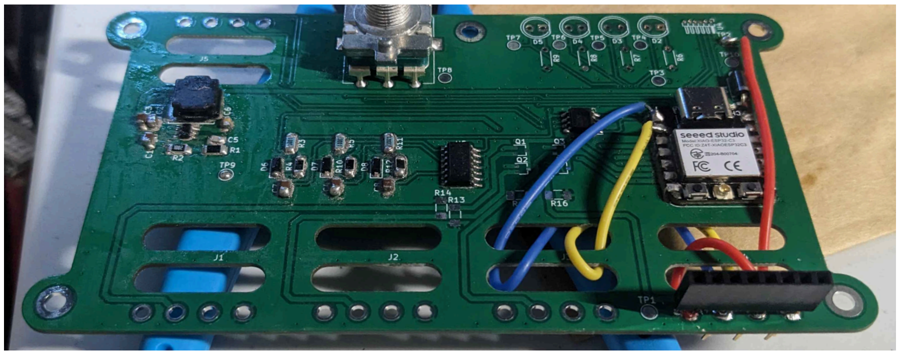


Fig 9: Modified assembly of Control Unit PCB (UART jumpers and LEDs not attached)

One final note on the hardware prototype is that there were issues with the encoder, which led to significant changes to the debouncing logic. Initially, it was planned for there to be hardware debouncing, as interrupt-based software was being used. However, the output from the encoder was found to not follow proper quadrature encoding, as specified in its datasheet (Fig. 10). Since both encoders exhibited spurious behavior, it was concluded that this issue was due to manufacturing defects, and so the hardware debounce was removed to simplify a software workaround.

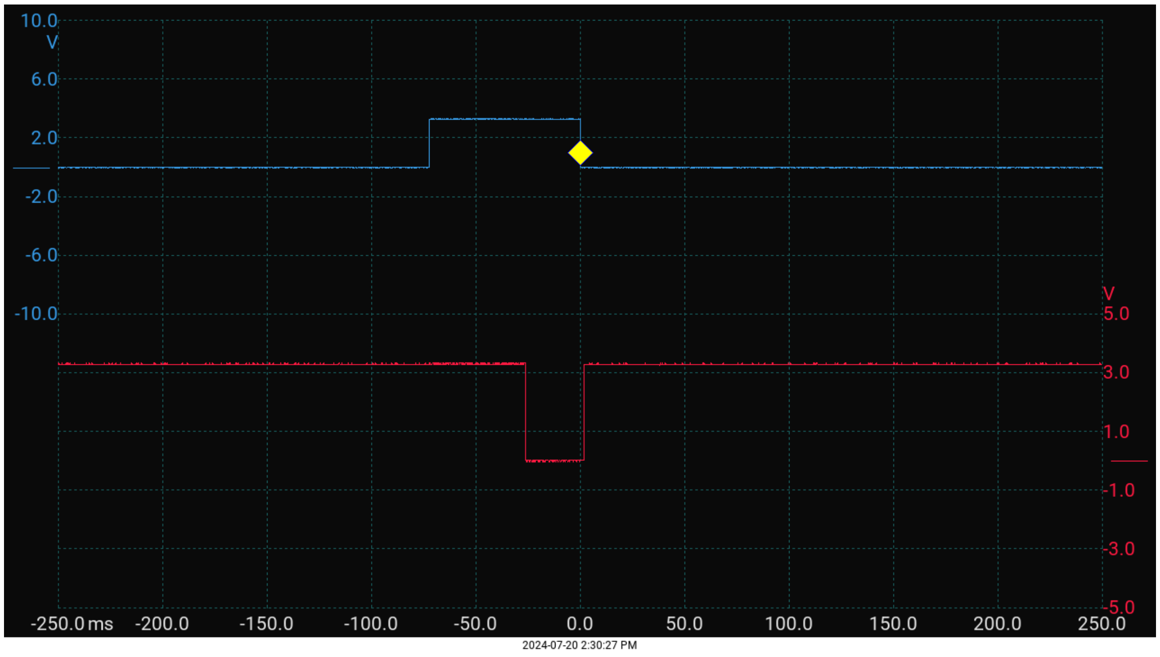


Figure 10: Example of one incorrect output from the encoder (both signals should be pulled high, drop low, then return to high; only signal B behaves correctly)

Enclosure Prototype

To house all of the components of both the command unit and the actuator unit, an enclosure needed to be designed and 3D printed. To accomplish this, Autodesk Fusion360 was the software used. All of the components were dimensioned, and using KiCAD's .STEP export feature a 3D model of the PCB was used to plan the layout of the chassis. The chassis was designed as two separate pieces, a lid and a body. These are attached together with four corner screws.

Each enclosure has multiple holes in it; these are four holes for the valve cables, four holes for the debug LEDs to monitor the status of the valves, and one hole for the CAN connection between devices. For this prototype, we did not have time to acquire Deutsch connectors to use for more professional cable connections, so simple filleted holes were used instead.

On the control unit, the enclosure is taller and the lid has additional cutouts. These are for the rotary encoder and the LCD; the LCD is mounted to the lid itself, whereas the encoder is connected to the PCB and has a simple knob attached to it. Fig. 11 shows this enclosure design. For the actuation unit, the height can be much shorter (since much of the height in the control unit is due to the LCD and encoder), thus the entire unit is a bit smaller. In Fig. 12, the actuation unit is actually upside down compared to the control unit due to the stiff cables which were used for the connection between the boards.

For the control unit enclosure, the lid ended up with a minor redesign due to some measurement error in the original design. The control unit was 3D printed first, and notes were made on issues in the design which were attempted to be corrected on the actuation unit; namely, there were issues with the original design of the PCB mounting hardware. This was adjusted on the actuation unit, but the new inserts for standoffs also did not work as planned; the enclosure would require at least one more iteration.

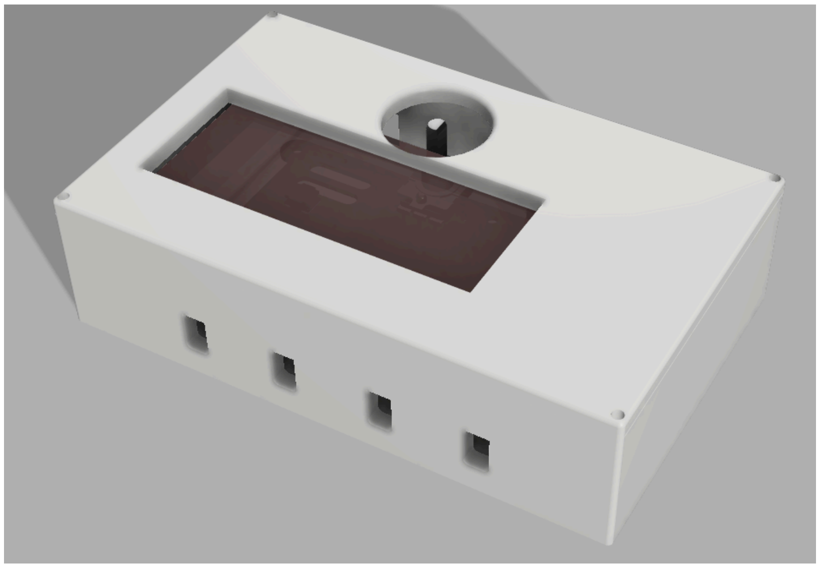


Figure 11: 3D model of enclosure before fabrication.

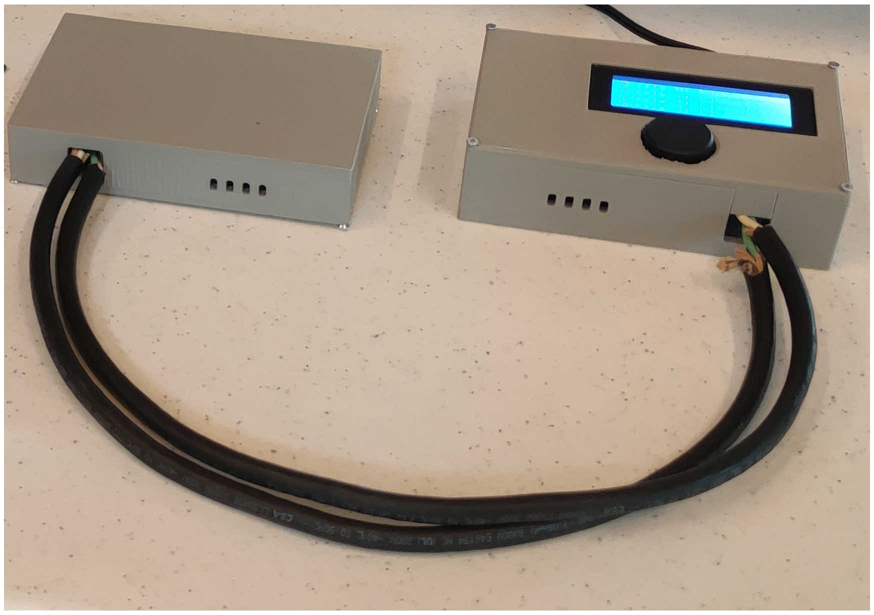


Figure 12: The final assembled unit presented for public viewing.

Software Prototype

General Overview

The overall structure of prototype software, that is, two separate pieces of firmware on two separate MCUs, remained the same as the initial specification. However, in order to produce a working prototype in time for the presentation, some changes and omissions occurred in the creation of the prototype software.

The main difference from specified to prototype software was the omission of the AgIsoStack++ library. While the library was compatible with the MCUs and was behaving as expected when tested, it was ultimately stripped out of the final prototype software due to one of the physical Canbus modules failing. Had the Canbus module not failed, it was likely that AgIsoStack++ would have been present in the final prototype.
With the omission of AgIsoStack++, an alternative communication protocol between the command and actuator units had to be used. After some research, UART was chosen as the replacement protocol, as the MCUs powering the units supported UART communication with no additional hardware. By reusing the planned Canbus pins, UART communication was successfully implemented on both the command and actuator units. The UART communication solution worked by sending a single number, either 0, 1, 2, or 3, from the command unit to the actuator unit based on the position of the rotary encoder at the time the integrated button was pressed. The actuator unit would receive the number, and based on the number it received, would toggle an LED state, which represented a nozzle state change.

Command Unit Software

While the command unit software's function matches that described in the specification, the software underwent a full rewrite during the completion of the prototype.

Initially, it was planned for the command unit's software to be written using the ESP-IDF programming framework. During the prototyping phase, though, a major issue was encountered with the selected rotary encoder, as it exhibited unintended behaviour on rotation. Despite trying alternative rotary encoder libraries, the encoder's input to the command unit was still misbehaving. In order to gain access to alternative rotary encoder libraries, the decision was made to rewrite the command unit software using the Arduino programming framework. The Arduino framework is implemented in C++, the same as ESP-IDF, which meant that a significant amount of code could be reused during the rewrite. To have parity with the initial command unit featureset, alternative libraries for both the LCD and rotary encoder were sought out. The LCD library was changed to "LiquidCrystal_I2C" [21], while the rotary encoder library was changed to "RotaryEncoder" [22]. The new rotary encoder library managed to mitigate the issues faced by allowing us to set a "latch" position, which was modified to match the behaviour of our rotary encoder.

Actuator Unit Software

The function of the actuator unit software remained the same as specified: receive a command from the command unit and act upon that command. The way in which the actuator unit received and processed commands did change, though.

Following the pivot to UART communication relatively late in the design process, the way the actuator unit received commands was forced to change. To simplify the code, both threads from the original actuator unit software were stripped out and replaced with a single process. This process would constantly listen for a UART transmission from the command unit, and once received, the command would immediately be acted upon. In the interest of time, reintroducing threads and the FIFO queue from the original software was not completed.

VIII Testing & Validation

In the completion of the prototype, multiple hardware and software tests were completed to ensure correct functionality. A summary of these tests can be seen below in Table 1, while the full test case document is available in Appendix C.

Table 1: Summarized test case table

Test No.	Description	Result
H1	Confirm Regulator Functionality	PASS
H2	Confirm CANBUS functionality	FAIL
S1	Confirm external hardware communication with command unit	PASS
S2	Confirm AgIsoStack++ compatibility with microcontrollers	PASS
S3	Confirm UART communication between command and actuator unit	PASS
S4	Confirm correct rotary encoder behaviour	FAIL

IX Cost Analysis

A full cost analysis of the project can be found in this section. All costs mentioned are in Canadian dollars unless otherwise specified.

The enclosure print cost totaled $28, summing from $8 of material used (256g) and $20 of labour.

The prototype PCB costs are summarized in Table 2, totalling $174.74:

Table 2
Summary of Invoiced Hardware Expenses

Invoice	Purpose	Cost
Digikey 104976436	Encoders	$19.71
Amazon CA436GF1W3KI	LCDs	$37.96
JLC 6699332A202406160315214	PCBs	$11.26
Digikey 105396631	Parts	$72.47
FedEx 2-628-74008	Brokerage Fee	$33.34

Labour costs are summarized as follows, assuming a regular hourly rate of $29.32 per member based on the average salary of a Junior Engineer of 61k/year [23] assuming 40 hours per week, and overtime (OT) as defined in [24] (Tab. 3):

Table 3
Breakdown of Engineering Hours

Member	Reg. Hours	OT Hours	Cost
Benjamin Lyne	52.5	0	$1539.30
Elliott Starchuk	40	0	$1172.80
Jensen Gillett	30	0	$879.60

Transportation, meal, and other direct costs not otherwise specified were minimized throughout the project due to a strong work-from-home ethic, and thus are zero. Likewise, careful planning allowed indirect costs not specified to also be negligible.

Therefore, the total expenditures for this project was $3794.44. The overall mechanical cost breaks down to $202.74 for a pair of Command and Actuator units, which includes shipping, brokerage, and manufacturer's fees, and $3591.70 of engineering hours. For a full 60-nozzle system, 15 Actuator Units and one Control Unit are needed. Since they are roughly the same design (with some minor component changes), the full system is estimated to cost 16 x $202.74 / 2 = $1621.92 to manufacture, not including nozzles.

Funding was provided by the University of Victoria, and CEWIL. UVic provided $120, while CEWIL provided $750 ($250 per student). This brings total expenditures down to $2924.44.

At a ~$1600 figure, the prototype is within the manufacturing and parts budget of $4000. Using the retail price of the Pro Stop Air Nozzle Body DCV from [25] as an estimate, 60 nozzle would cost $1440, for a total system price of $3061.92. PCB assembly (PCBA) is assumed to be $9.30 per board, based on rates and a quote from JLCPCB [18], increasing the manufacturing cost by $558 to $3619 (which is then rounded up to $3700). It is expected that another iteration, both hardware and software is needed, so $3800 is added twice to the ROI calculation (rounded up from the total expenditures discussed above). It is assumed for this calculation that the manufacturing price would not change (see below for a discussion on areas of cost saving). Marketing is the other major expenditure, estimated at $1000 per month for four months. The income is the expected maximum retail price of $12000 per sale, which can be adjusted once final engineering and marketing expenses are better known, at the abysmal rate of one sale per month. See Tab. 4 for a breakdown of costs and projected income, detailing a rapid ROI once the product hits market, or, more realistically, that the retail price could be significantly dropped (pending accurate figures for marketing, prototype 2, etc.).

Table 4
Projected Costs and Income

Item	**Date(s)**	**Income**	**Expense**
Prototype 1	May ‘24 - July ‘24		$3800
Prototype 2	Aug ‘24 - Oct ‘24		$3800
UVic Funding	Aug ‘24	$120	
CEWIL Funding	Aug ‘24	$750	
Manufacturing/Parts	Nov ‘24		$3700
Marketing	Aug ‘24 - Nov ‘24		$4000
Sales	Dec ‘24 - Jan ‘25	$24000	
Totals		$24870	$15300
Profit (10 months)	$9570		
	ROI is achieved in 10 months		

Finally, there are a number of places for cost savings. Primarily, it is of note that a large portion of the expenditures in Tab. 2 actually came from shipping costs. The chosen supplier, DigiKey, waives this cost for sales of over $100, so it is expected to save around $60 on parts. The larger savings would come from purchasing electronic components in bulk, which has 30%-50% savings possible depending on quantities purchased.

X Conclusion & Recommendations

Overall, we had to narrow the scope of this project from its original intention due to time limitations, and we did not meet all the original objectives of the project. This includes changing the communication mechanism from Canbus to UART, as well as not interacting with a physical nozzle.

This design does however provide a good base to work from, for a future iteration that could be used for its intended deployment in the field. Future work includes revising the CAN connectivity to fix communications between the boards, alongside testing with multiple boards, and eventually full integration with an existing RCU.

References

[1] M. Wengryn, private communication. May 2024.

[2] "Code of ethics," egbc.ca. Accessed Jun. 23, 2024. [Online]. Available: https://www.egbc.ca/complaints-discipline/code-of-ethics/code-of-ethics.

[3] *Failure Mechanism Based Stress Test Qualification For Integrated Circuits In Automotive Applications*, AEC-Q100, Rev. J, Automotive Electronics Council, 2023. Accessed Jul. 11, 2024. [Online]. Available: http://www.aecouncil.com/Documents/AEC_Q100_Rev_J_Base_Document.pdf.

[4] M. Sima, private communication. July 2024.

[5] *Product Catalog*, no. 33, KZValve, 2024, pp. 62-76.

[6] *Pentair - Hypro Retail Price List*, Farmco, 2022, p. 28. Accessed June 13, 2024. [Online]. Available: https://www.farmco.com/price%20lists/hypro/2022%2003-14%20hypro%20pentair.pdf.

[7] "Seletron complete system for AG sprayers (Bravo 400S).", partstospray.com. Accessed Jun. 13, 2024. [Online]. Available: https://partstospray.com/seletroncompletesystemforagsprayers.aspx.

[8] *Prostop-E ISOBUS Module System*, HYP1100, Pentair, 2021, p. 5. Accessed Jun. 21, 2024. [Online]. Available: https://www.manualslib.com/manual/2203932/Pentair-Prostop-E.html?page=5#manual.

[9] *High-Speed CAN Transceiver with Standby Mode CAN FD Ready*, ATA6560/1, no. DS20005991B, Microchip Technology Inc., 2019. Accessed Jun. 18, 2024. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/20005991B.pdf.

[10] "Seeed Studio XIAO ESP32C3 - RISC-V tiny MCU board with Wi-Fi and Bluetooth5.0, battery charge supported, power efficiency and rich interface," seeedstudio.com. Accessed Jun. 18, 2024. [Online]. Available: https://www.seeedstudio.com/Seeed-XIAO-ESP32C3-p-5431.html.

[11] *4.2V to 18V Input, 2A Low IQ Synchronous Buck Converter*, AP62200/AP62201/AP62200T, no. DS41957, Rev. 6-2, Diodes Inc., 2023. Accessed Jul. 11, 2024. [Online]. Available: https://www.diodes.com/assets/Datasheets/AP62200_AP62201_AP62200T.pdf.

[12] *Xiao ESP32C3*, Rev. 1.0, Seeed Studio, 2022. Accessed Jun. 11, 2024. [Online]. Available: https://files.seeedstudio.com/wiki/XIAO_WiFi/Resources/Seeeduino-XIAO-ESP32C3-SCH.pdf.

[13] *High Current, High Speed LDO Regulators*, XC6210 Series, No. ETR0317_007, Torex Semiconductor Ltd. Accessed Jul. 11, 2024. [Online]. Available: https://product.torexsemi.com/system/files/series/xc6210.pdf.

[14] "Logic gates," in *Lessons in Electric Circuits*, vol. Digital Circuits, allaboutcircuits.com. Accessed Jun. 18, 2024. [Online]. Available: https://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/.

[15] *N-Channel Enhancement Mode MOSFET*, DMN2004WK, no. DS30934, Rev. 6-2, Diodes Inc., 2024. Accessed Jun. 18, 2024. [Online]. Available: https://www.diodes.com/assets/Datasheets/DMN2004WK.pdf.

[16] J. G. Ganssle, "A guide to debouncing," The Ganssle Group, Baltimore, MD, USA, Rev. 3, 2008. [Online]. Available: https://my.eng.utah.edu/~cs5780/debouncing.pdf.

[17] "PCB trace width calculator." digikey.com. Accessed Jun. 18, 2024. [Online]. Available: https://www.digikey.ca/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width.

[18] "PCB manufacturing & assembly capabilities." jlcpcb.com. Accessed Jun. 18, 2024. [Online]. Available: https://jlcpcb.com/capabilities/Capabilities.

[19] A. Del Grosso *et al., AgIsoStack++*. Open-Agriculture/AgIsoStack-plus-plus. (2024). Open Agriculture. Accessed May 16, 2024. [Online]. Available: https://github.com/Open-Agriculture/AgIsoStack-plus-plus.

[20] A. Del Grosso *et al.,* "AgIsoStack++." agisostack-plus-plus.readthedocs.io. Accessed May 16, 2024. [Online]. Available: https://agisostack-plus-plus.readthedocs.io/en/latest/.

[21] J. Rickman *et al., LiquidCrystal_I2C.* johnrickman/LiquidCrystal_I2C. (2016). Accessed July 27, 2024. [Online]. Available: https://github.com/johnrickman/LiquidCrystal_I2C.

[22] M. Hertel *et al., RotaryEncoder.* mathertel/RotaryEncoder. (2022). Accessed July 27, 2024. [Online]. Available: https://github.com/mathertel/RotaryEncoder.

[23] "Junior Engineer Salaries in Canada," Glassdoor. Accessed Aug. 04, 2024. [Online]. Available: https://www.glassdoor.ca/Salaries/junior-engineer-salary-SRCH_KO0,15.htm.

[24] Overtime Wages for Employees Not Working Under an Averaging Agreement - Act Part 4, Section 40," Government of British Columbia. Accessed Aug. 04, 2024. [Online]. Available: https://www2.gov.bc.ca/gov/content/employment-business/employment-standards-advice/employment-standards/forms-resources/igm/esa-part-4-section-40

[25] "Pro Stop Nozzle Body DCV," Nozzle Ninja. Accessed Aug. 04, 2024. [Online]. Available: https://nozzleninja.com/products/pro-stop-air-nozzle-body-dcv .

Appendix A: EGBC Code of Ethics

The EGBC Code of Ethics is contained in this appendix. This project fully adhered to the code during its completion. The Code of Ethics was directly sourced from EGBC [2].

Registrants must act at all times with fairness, courtesy and good faith toward all persons with whom the registrant has professional dealings, and in accordance with the public interest. Registrants must uphold the values of truth, honesty, and trustworthiness and safeguard human life and welfare and the environment. In keeping with these basic tenets, registrants must:

1. Hold paramount the safety, health, and welfare of the public, including the protection of the environment and the promotion of health and safety in the workplace;
2. Practice only in those fields where training and ability make the registrant professionally competent;
3. Have regard for the common law and any applicable enactments, federal enactments, or enactments of another province;
4. Have regard for applicable standards, policies, plans, and practices established by the government or Engineers and Geoscientists BC;
5. Maintain competence in relevant specializations, including advances in the regulated practice and relevant science;
6. Provide accurate information in respect of qualifications and experience;
7. Provide professional opinions that distinguish between facts, assumptions, and opinions;
8. Avoid situations and circumstances in which there is a real or perceived conflict of interest and ensure conflicts of interest, including perceived conflicts of interest, are properly disclosed and necessary measures are taken so a conflict of interest does not bias decisions or recommendations;
9. Report to Engineers and Geoscientists BC and, if applicable, any other appropriate authority, if the registrant, on reasonable and probable grounds, believes that:
    a. The continued practice of a regulated practice by another registrant or other person, including firms and employers, might pose a risk of significant harm to the environment or to the health or safety of the public or a group of people; or
    b. A registrant or another individual has made decisions or engaged in practices which may be illegal or unethical;
10. Present clearly to employers and clients the possible consequences if professional decisions or judgments are overruled or disregarded;
11. Clearly identify each registrant who has contributed professional work, including recommendations, reports, statements, or opinions;
12. Undertake work and documentation with due diligence and in accordance with any guidance developed to standardize professional documentation for the applicable profession; and

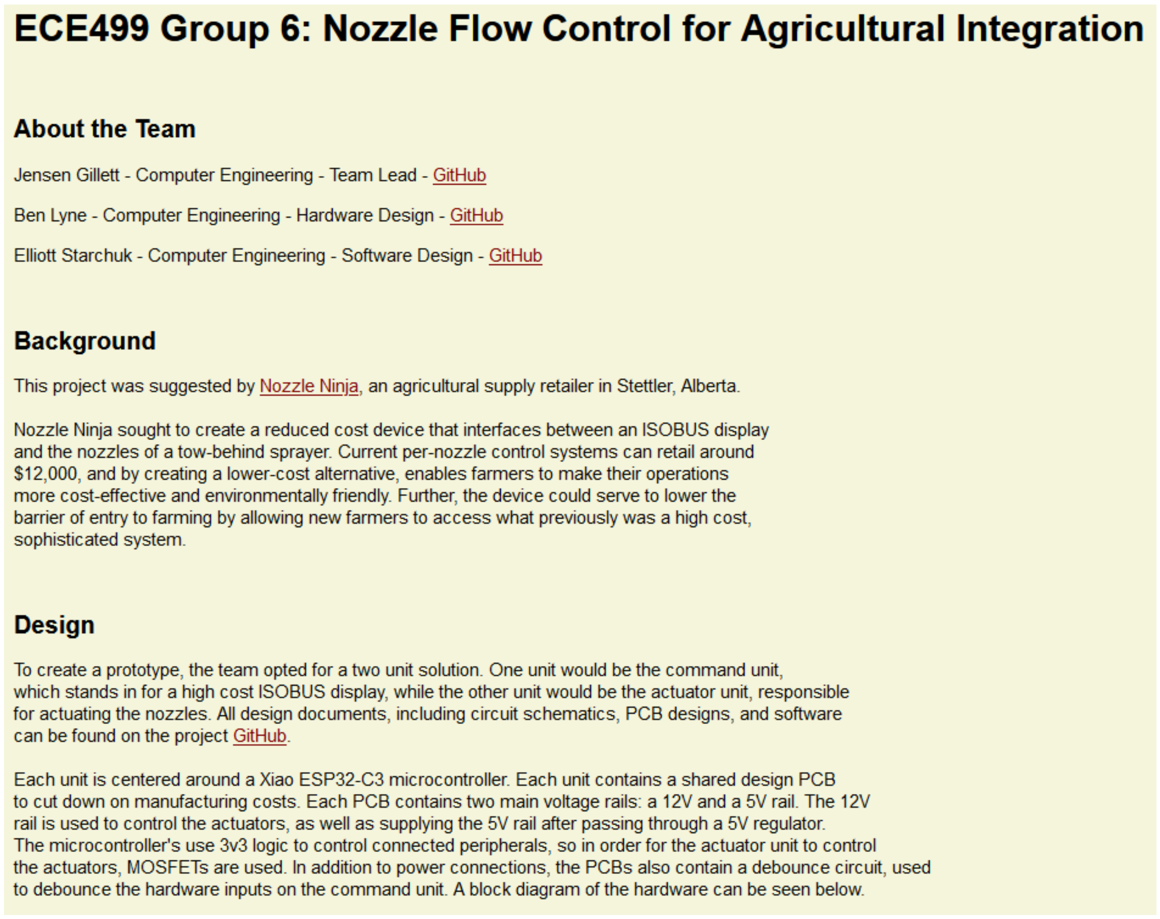
13. Conduct themselves with fairness, courtesy, and good faith towards clients, colleagues, and others, give credit where it is due and accept, as well as give honest and fair professional comment.

Appendix B: Website

The website for this project can be found at https://tropingenie.github.io/ECE-499-Website/.

It is designed to look similar to most of the ECE department webpages, but is written in modern compliant HTML5.

A screenshot of the main page of the website can be seen below in Figure B1.



ECE499 Group 6: Nozzle Flow Control for Agricultural Integration

About the Team

Jensen Gillett - Computer Engineering - Team Lead - GitHub

Ben Lyne - Computer Engineering - Hardware Design - GitHub

Elliott Starchuk - Computer Engineering - Software Design - GitHub

Background

This project was suggested by Nozzle Ninja, an agricultural supply retailer in Stettler, Alberta.

Nozzle Ninja sought to create a reduced cost device that interfaces between an ISOBUS display and the nozzles of a tow-behind sprayer. Current per-nozzle control systems can retail around $12,000, and by creating a lower-cost alternative, enables farmers to make their operations more cost-effective and environmentally friendly. Further, the device could serve to lower the barrier of entry to farming by allowing new farmers to access what previously was a high cost, sophisticated system.

Design

To create a prototype, the team opted for a two unit solution. One unit would be the command unit, which stands in for a high cost ISOBUS display, while the other unit would be the actuator unit, responsible for actuating the nozzles. All design documents, including circuit schematics, PCB designs, and software can be found on the project GitHub.

Each unit is centered around a Xiao ESP32-C3 microcontroller. Each unit contains a shared design PCB to cut down on manufacturing costs. Each PCB contains two main voltage rails: a 12V and a 5V rail. The 12V rail is used to control the actuators, as well as supplying the 5V rail after passing through a 5V regulator. The microcontroller's use 3v3 logic to control connected peripherals, so in order for the actuator unit to control the actuators, MOSFETs are used. In addition to power connections, the PCBs also contain a debounce circuit, used to debounce the hardware inputs on the command unit. A block diagram of the hardware can be seen below.

Figure B1: Screenshot of the main page of the website.

Appendix C: Test case document

Test Suite	Test No.	Description	Interface	Pre-requisites
Unit testing - hardware	H1	Confirm Regulator Functionality	Hardware	Regulator, Microcontroller
Unit testing - hardware	H2	Confirm CANBUS functionality	Hardware-software	Microcontroller, CAN Transceiver
Unit testing - software	S1	Confirm external hardware communication with command unit	Hardware-software	Microcontroller Rotary encoder LCD LEDs
Unit testing - software	S2	Confirm AgIsoStack++ compatibility with microcontrollers	Software	Microcontroller
Unit testing - software	S3	Confirm UART communication between command and actuator unit	Software	Microcontroller (2)
Unit testing	S4	Confirm correct rotary encoder behaviour	Hardware-software	Microcontroller Rotary encoder

Test No.	Test Steps	Expected results	Actual results	RESULT
H1	Apply 12V to power input, Observe 5v and 3v3 test points	Test point voltages should be within 5% of nominal	As expected	PASS
H2	Power on board with AgIsoStack++ test code Send commands by manipulating the Command Unit Observe microcontroller state via the LEDs	Actuator unit should mimic the Control Unit exactly	Actuator unit dies when connected to Command Unit, CAN Transceiver shorts to ground	FAIL
S1	Initialize microcontroller communication with external peripherals Push output to LCD Accept input from rotary encoder and observe on console Toggle LED state once per 5 seconds	LCD: expected output appears Rotary encoder: software count increases when rotated CW, decreases CCW LED: state toggles in accordance with the code	LCD: as expected Rotary encoder: counter variable shows unexpected behaviour, but MCU can receive communication LED: as expected	PASS
S2	Initialize microcontroller and AgIsoStack++ following the tutorial Compile code Observe microcontroller state via the serial monitor	Code with AgIsoStack++ integration compiles successfully Microcontroller does not crash/hang when running the compiled code	As expected	PASS
S3	Initialize microcontrollers with UART communication code Have one MCU transmit, the other receive View console and ensure UART transmission is successfully received	Console shows transmitted UART message	As expected	PASS
S4	Initialize microcontroller with rotary encoder test code Rotate encoder both clockwise and counterclockwise View console and watch the counter variable	Counter variable increases by one with each CW detent Decreases by one after each CCW detent	Counter variable shows unintended behaviour (increasing on CCW, skipping steps, etc.)	FAIL