Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #48655

WIP-Experimentation in control and communication methods for neuron-based adaptable robotics

Dr. Iftekhar Ibne Basith, Sam Houston State University

Dr. Iftekhar Ibne Basith is an Associate Professor in the Department of Engineering Technology at Sam Houston State University, Huntsville, TX, USA. Dr. Basith has a Ph.D and Masters in Electrical and Computer Engineering from University of Windsor, ON,

Syed Hasib Akhter Faruqui, Sam Houston State University

Assistant Professor, Department of Engineering Technology

Michael Tyler Johnson-Moore, Sam Houston State University

WIP: Experimentation in control and communication methods for neuron-based adaptable robotics

Michael Johnson-Moore, Syed Hasib Akhter Faruqui and Iftekhar Ibne Basith
Department of Engineering Technology, Sam Houston State University, Huntsville, TX

ABSTRACT

This paper discusses the partial development of a modular robotic arm built around a physical bidirectional tree-like architecture inspired by artificial neural network (ANN), intended for use in experimentation with control and communication methods. This project when completed will provide students with foundation and experience in developing modular robotics and ANN controller with the long-tern goal of developing smart prostheses. The primary design goal was to develop an adaptable robotic system capable of responding to sensor data and avoiding obstructions through a distributed network of processing nodes. This manuscript describes the framework for several communication methods including traditional networking protocols using the IEC 61850 standard or onion routing, mathematical transformation-based routing, and neural network approaches while maintaining safety. However, due to time constraints at the time of this publication the project is still in work-in-progress status and needs further time to implement and validate the proposed methodology. The hardware implementation encountered multiple design and manufacturing issues that are documented as lessons learned in this manuscript. This project provides insights into the challenges of developing complex robotic systems with distributed control architectures and serves as a foundation for future work in this domain.

This project was implemented as part of the SHSU ETEC 4199 and ETEC 4399 senior design courses, which assess several student learning outcomes related to ABET-ETAC and ATMAE standards.

I. INTRODUCTION

This document discusses the construction of a robotic human arm using a novel internal architecture inspired by the human nervous system. The focus of this project was on the internal electronic and communications systems of the arm, not on mechanical capabilities.

Currently, human prostheses are based on traditional robotics and controlled indirectly by taking advantage of conscious muscle movement with little to no effect on other parts of the body, such as moving ears or wrist muscles or flexing muscles to give commands to the robotic prosthesis. This approach, known as myoelectric control, uses electromyogram (EMG) signals recorded from the patient's residual muscles, which are then processed and used as control inputs to drive motors coupled to the prosthetic limb [1]. Some research has been done to directly read intention from brain scans, but this requires either an invasive brain implant, or that the patient to be placed in large machines with many probes reading the voltage of surface nerves of the skull. Non-invasive BCI approaches typically use electroencephalography (EEG), where topical electric sensors are placed over the head to measure brain activities [2]. However, these systems are limited to simple commands such as up or down and require extensive training before the patient and computer find

a pattern of thinking that works. Although non-invasive scalp recorded EEG signals can be used successfully to manipulate prosthetic devices, the control is still limited compared to other methods [3]. Recent developments by Neuralink appear to have recently made significant progress in intention interpretation [4].

Ideally, direct neural integration between a prosthetic device and the patient's original nervous system would provide intuitive control that mimics natural limb functionality [5]. However, due to access limitations of an undergraduate program and challenges involving such neural interface technology this is outside the scope of this time-limited undergraduate engineering project.

It is a reasonable assumption that a robotic arm interfacing with the human nervous system might benefit from an engineering architecture based on the human nervous system. In recent years, such biomimetic approaches have demonstrated significant efficacy in machine learning applications where neural interfaces developed for prosthetic control aim to establish more natural and intuitive connections between the nervous system and artificial devices [6], further supporting the assumption. This assumption, while plausible, requires empirical validation; thus, this project is designed to provide a means of testing this hypothesis. This engineering experiment has two broad qualitative outcomes:

1. The arm does not work as intended; it is unable to self-direct, adapt to changes, or is not sufficiently responsive.
2. The arm works as intended; in which case a variety of quantitative performance measurements can be made to determine optimal design guidelines for related projects.

A. Goals of this Project

The primary objective of this research was to develop a generic robotic system with adaptive capabilities that can respond to a range of physical geometries, potentially including self-modification. This system is intended to serve as a foundation for future development of neural interfaces for smart prostheses or other brain-computer integration technologies, while simultaneously providing the researcher with valuable experience in robotics and artificial neural network based controls.

An ideal prototype would demonstrate the following capabilities:

- Provide the operator with the ability to construct a robot as needed from individual limbs, without requiring explicit programming of geometric parameters.
- Self-adapt to changes in geometry, without user, utilizing integrated sensors to detect its own environment, identify potential obstructions, and determine safe movement ranges; and
- Execute high-level commands such as object retrieval without requiring explicit path planning, demonstrating the ability to navigate environmental obstacles autonomously as a part of this goal.

II. DEFINITION OF TERMS

To prevent confusion, unless otherwise specified, the following terms are used according to these definitions throughout this document. All monetary values are in United States Dollars (USD), unless otherwise specified.

Limb – a complete, fully constructed arm segment or joint, which contains modular attachment point(s) for expansion, and consists of one or more modules.

Module – an organizational concept comprised of purpose-built hardware containing one or more neurons.

Nerve, neuron, or **node** – a software or mathematical construct, representing a single node in an artificial neural network (ANN). When referring to the biological concept, the term is prefaced with "biological."

Component (outside circuit board context) – a limb, module, or neuron, determined by context.

Component or **part** (in context of a circuit board) – an integrated circuit chip or discrete electronic device soldered onto the circuit board after manufacture.

Segment – a limb or module, determined by context.

Proximal is a medical term meaning "towards the direction of the attachment point of a limb." [7] For this project, it means "In the direction of the central controller." Distal refers to the opposite direction.

III. Engineering Methodology

The initial phase of this project involved designing and constructing the robotic arm. To ensure sufficient time for manufacturing, this was completed prior to developing the simulated version. Additionally, the necessity of having both the prototype and simulation available for testing dictated that both components be developed before conducting any experimental analysis. In this Work-in-Progress (WIP) project we were able to complete only the hardware design and partial construction of the robotic arm modules, while the firmware development, simulation environment, and testing phases remained unfinished due to time constraints and project management challenges. Thus, no definite quantitative result will be presented in this work.

A. Testing protocols

The experimental protocol consisted of giving the simulation or arm a standardized set of benchmarking commands for each communication method described below in § III.D. Due to the incomplete state of communication programming and the lack of finalized command specifications, these testing standards were not established as of publication. Furthermore, sensor calibration procedures were not performed.

The planned steps for the project would look as follows:

1. Program the logic of each communication protocol into the simulator. Verify full functionality within the 2D environment. If a protocol demonstrates inoperability and cannot be modified to achieve full functionality, designate it qualitatively as nonviable. Both algorithmic and command list modifications would be permissible during this stage.

2. Implement the finalized logic into the arm's firmware and test for basic functionality. If necessary, return to step 1 for further adjustment.
3. Iterate this process until all desired protocols are properly configured.
4. Upon completion of programming for all desired protocols, repeat steps 1-3 to standardize command lists and functional features across all protocols if any modifications were introduced.
5. In the simulation environment, quantitatively assess each protocol for latency in message propagation between the transmission of an unsafe command and the arm's response to detected obstructions.
6. (Limited single communication protocol) Within the simulation, the arms would be tested with several random movement commands and randomly placed obstructions, quantitatively measuring the average and maximum message count between simulated neurons, the average and maximum message size and bandwidth requirements, prevented collisions per thousand move commands, and unprevented collisions per thousand move commands. The number of random commands would start at 1000 and increase until the results stop significantly changing from the previous tests.

7. Step 6 would be repeated with varying arm lengths and geometries. Parameters for testing would include the arm's total number of joints, frequency of branching, and sensor density between joints.
8. The data from step 6 would be used to calculate a maximum safe angular speed for each joint of the constructed prototype. The simulation and proposed command list would be matched to the actual prototype's geometry to verify safety.
    a. In the event of a collusion, step 8 would be repeated with a reduced joint movement speed. If collisions remain unavoidable, the communication protocol would be qualitatively marked nonviable.
    b. In the absence of collision, the command list would be tested on the actual arm as both verification and demonstration purposes.
9. In the absence of collision, the protocol would be tested with live arbitrary movement commands and user-re-arrangeable obstructions to ensure operational safety in a production environment.
10. Steps 6-9 would be repeated for each communication protocol under evaluation.

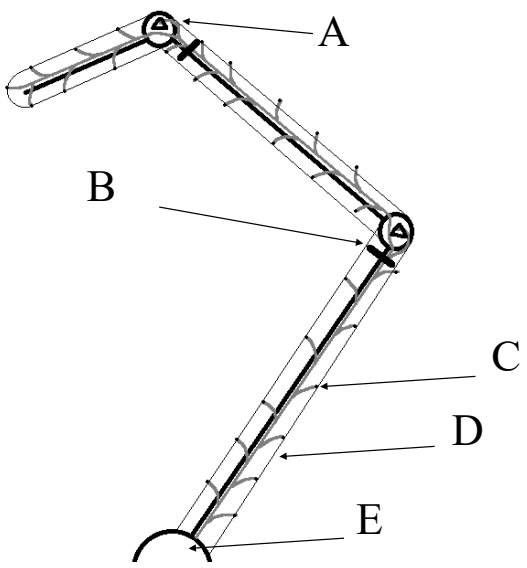

Figure 1. An illustration of a basic geometry for an arm following this project's design. Actual sensor density should be much higher. **A**: 2x joint module, including joint control neuron and motor; **B**: 2x reflex controller module, containing a reflex controller neuron; **C**: 29x sensor module, containing inductive and capacitive collision sensors, sensor neurons, and communication neurons; **D**: 3x arm limb; **E**: controller module, containing central control neuron and optionally a base joint module.

B. General design overview

Physically, the arm is constructed from a series of modular limbs, providing a chain of modules throughout the arm, serving various purposes. An example arm showing limbs and modules is shown in Figure. 1. Each module consists of, and communicates with other modules through, a branched chain of simulated "neurons," as shown in Figure. 2, forming a physical artificial neural network extending throughout the arm. This biomimetic approach draws inspiration from brain-inspired control techniques that closely emulate motor functions based on current neuroscientific insights. Although inspired by the human nervous system, these are not based on human neurons. Although the neurons are labeled by their function, these are annotative labels; the neurons are built using a single generic programmed system with specialization as needed. This design philosophy aligns with approaches where biologically realistic robots are controlled by spiking neural networks that mimic brain mechanisms [8].

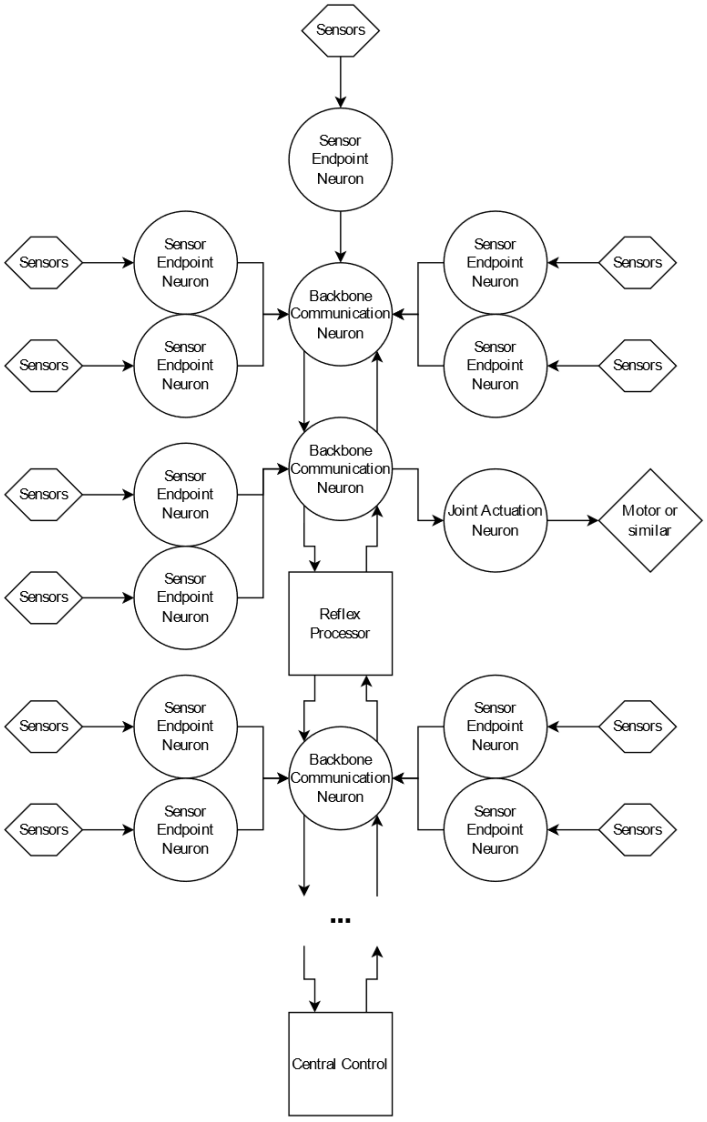

Figure 2. Example of a possible neuron chain. Two component types, the central controller and reflex processors, are marked using squares to indicate that they might require more computing power than the other neurons.

C. Component types

The module and neuron components are organized into five general types (communication, central control, reflex processing, joint actuation, and sensor), based on function. Modules contain at least one neuron of the same type as themselves in order to gain that functionality. A brief overview of each component type is provided in Table I.

Table 1. Quick Reference Table of Component Types

Name	**Function**

Central Controller	Commands arm, learns geometry, learns environment.
Communication	Links components together. Central part of each other type.
Sensor Endpoint	Reads sensors, forwards values through communication backbone.
Reflex Processor	Responds to sensor readings, modifying given commands to avoid collision.
Joint Actuation	Controls the joints according to received commands.

1) Communication Neurons

Communication Neurons take in a set of values from inputs, modify them as needed, and pass them to outputs. Inputs can be received from the previous communication neuron or generated internally (such as sensor data). Outputs may be values which get passed along to the next communication neuron, or functions handled internally (such as actuating a joint).

The communication neurons in Figure 2 are labeled "Backbone Communication Neurons" because they form a channel along which all signals travel. Specialized limbs may contain communication neurons not part of the backbone if needed.

It is important to note that all neurons within the system are fundamentally communication neurons, with certain types possessing additional specialized functionalities. For example, a sensor endpoint neuron (below) programmatically starts as a communication neuron and performs the additional functionality of reading and passing sensor data. Because the arm's purpose is to provide a platform for experimentation with communication methods, the exact specification of communication neurons is left to the implementer. See below in III-C for more information on planned communication functionality.

2) The central control neuron

The Central Control Neuron, also referred to as the central controller, is the point from which control commands are issued and sensor data is ultimately sent. This is the "brain" of the arm, although unlike a human brain, it consists of a single highly capable neuron.

3) Reflex processing neurons

Reflex Processing Neurons are a safety feature intended to countermand centrally issued instructions if sensor data indicates the arm is about to collide with something. These are included to be able to respond faster than the central controller, with more specific reflex procedures than the generic central controller would be capable of. After handling sensor data, the reflex neuron's backbone passes it onward towards the central controller. If the reflex processor acted on the data, this must be noted somehow, in a manner determined by the communication method. If it did not, whether due to inability or lack of need, proximal reflex processors will get the opportunity to act.

4) The Joint Actuation and Sensor Endpoint Neurons

The Joint Actuation and Sensor Endpoint Neurons interface with non-neuron devices. Sensor endpoints interpret data from non-neuron systems and convert readings to the format used by the

neurons. Joint actuation neurons take control signals from proximal neurons, and translate those signals into a real-world effect, such as moving part of the arm. Every joint actuation neuron must be distal to a reflex processing neuron to accurately and safely receive reflex commands. There must be sensor neurons distal to the joint (not the neuron/module, the physical motor or other actuator) to provide data for the reflex neuron to act upon.

The standard organization of the sensor, reflex, and joint modules is provided in the appendix.

D. Hardware

The basic limb consists of a cylinder with a 3-inch diameter, constructed of 3-inch-long sensor modules (henceforth referred to as limb modules) organized according to Figure A1 in the appendix. A single limb module of the final prototype is shown in Figure 3, and the full KiCad design files are available at https://github.com/MichaelJ-SHSU/thesis-limb_module-rigid.

Figure 3. Proximal side of a single limb module, shown on its PVC rail.

Each limb module consists of a rigid central circuit board, with two flexible sensor boards connected to the edges and wrapped around to form a circle. All circuit boards were designed and manufactured for this project. The sensor board contains a checkerboard of 1 cm pads and coils for capacitive and inductive sensing. Each sensor board has 6 rows of 11 sensors, for a total of 132 sensor neurons per sensor module. PVC pipes, wooden dowels, and 3D-printed brackets provide rigidity throughout the limb, with a foam pad under the sensor boards to cushion impacts. The entire limb is wrapped in plastic wrap to prevent damage to the sensor boards. This arrangement is shown in Figure 4.

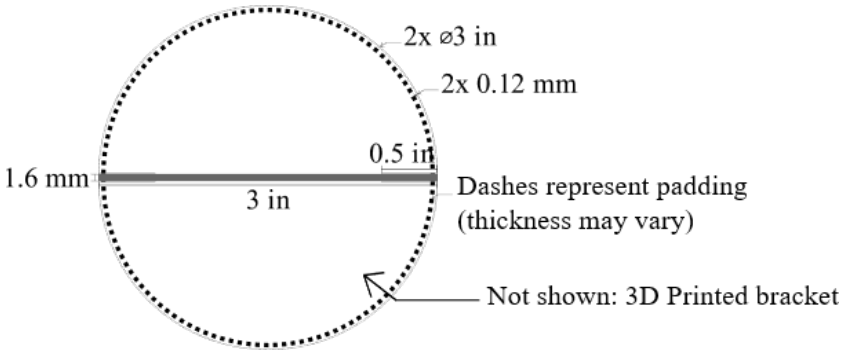

Figure 4. Cutout view of the limb.

The central board contains connections to adjacent modules, sensor controllers ICs, power regulation, and microcontrollers. The Raspberry Pi RP2040 microcontroller was chosen for its cheap price point of $1.00 [9], relatively high number of programmable IO pins, and large developer community. Each limb module has 2 sensor modules RP2040s and 1 communication backbone RP2040. An integrated USB 2.0 hub is included for ease of programming and to provide limited arm monitoring over USB.

Due to the time constraints the firmware for the RP2040s was not written. References to the firmware program in this chapter are the originally planned functionality.

For communicating between RP2040s, UART is used. For communicating with external devices, other protocols are available; the sensor controllers communicate over I2C and PIO I2C. The RP2040 includes two hardware I2C channels and two hardware UART channels. Because more channels are needed than exist, the remaining channels are added in software using programmable IO (PIO). UART channel 0 is used for debugging and UART channel 1 is used for communicating with the proximal RP2040. Communications with distal RP2040s must be done over PIO UART channels. One GPIO pin on each RP2040 is used for the detection interruption.

Using RP2040's 30 digital IO pins, minus 3 used for debugging and interrupt, this provides 13 potential two wire connections, or a maximum of 12 branches per RP2040. This provides the potential for a maximum theoretical limb module length of 24 inches, although sizes larger than 300 mm are prohibitively expensive to manufacture in small quantities. The formulae for this calculation is shown in Equation's 1-9. Whether the RP2040 has sufficient processing power to function in this scenario is another question.

$$R_s = R_b - 1 = 11 \tag{1}$$

Equation 1. Formula for calculating maximum sensor neurons for a non-terminal module, where R_b = maximum branches per RP2040 $= 12$ and R_s = maximum sensor neurons for a non-terminal module.

$$C_{I2C} = R_s R_b = 132 \tag{2}$$

Equation 2. Formula for calculating maximum number of possible I2C channels used for sensing in a fully utilized RP2040 based module, where R_b = maximum branches per RP2040 $= 12$, R_s = maximum sensor neurons for a non-terminal module, and C_{I2C} = maximum number of possible I2C channels used for sensing.

$$S_{eff} = \mathrm{lcm}(4,6) = 12 \tag{3}$$

Equation 3. Formula for calculating the number of sensors (of each type) available in efficient groups, for the sensor controllers listed below, where S_{eff} = the number of sensors (of each type) available in efficient groups, for the sensor controllers listed below. Four and six are the sensor channels available in LDC3114 and CAP1206, respectively.

$$k_{eff} = {}^{S_{eff}}/_{4} = 3 \tag{4}$$

Equation 4. Formula for calculating the number of I2C channels in each efficient group, where S_{eff} = the number of sensors (of each type) available in efficient groups, for the sensor controllers listed below, and k_{eff} = the number of I2C channels in each efficient group.

$$C_{effg} = \frac{C_{I2C}}{k_{eff}} = 44 \quad (5)$$

Equation 5. Formula for calculating the number of efficient groups of a fully utilized RP2040 based module, where C_{I2C} = maximum number of possible I2C channels used for sensing, k_{eff} = the number of I2C channels in each efficient group, and C_{effg} = the number of efficient groups.

$$S_{max} = 2C_{effg}S_{eff} = 1056 \quad (6)$$

Equation 6. Formula for calculating the maximum number of sensors possible if connected efficiently in a fully utilized RP2040 based module, where S_{eff} = the number of sensors (of each type) available in efficient groups, for the sensor controllers listed below, C_{effg} = the number of efficient groups, and S_{max} = the maximum number of sensors possible if connected efficiently.

$$L_{rows} = \left\lfloor \frac{S_{max}}{S_{row}} \right\rfloor = 48 \quad (7)$$

Equation 7. Formula for calculating the number of rows of sensors required, rounded down to not have sensor gaps, in a fully utilized RP2040 based module, where S_{max} = the maximum number of sensors possible if connected efficiently, S_{row} = the number of sensors in each row of the prototype (22), and L_{rows} = the number of rows of sensors required, rounded down to not have sensor gaps.

$$L_{inches} = \frac{L_{rows}}{L_{rows,p}} L_{inches,p} = 24 \quad (8)$$

Equation 8. Formula for calculating maximum theoretical length of a fully utilized RP2040 based module, where L_{rows} = the number of rows of sensors required, rounded down to not have sensor gaps, $L_{rows,p}$ = sensor rows on prototype = 6, $L_{inches,p}$ = length of prototype in inches = 3, and L_{inches} = maximum theoretical module length in inches.

The IC chosen for the capacitive sensor controller is the Microchip CAP1206. This controller was selected for its simple usage and 6-channel sensing capabilities [10]. Each 3-inch limb module uses 11 controllers.

For inductive proximity sensing, the Texas Instruments LDC3114 was chosen, selected because it has 4 channels, is provided in a package size with a manufacturable footprint, and supports raw data access. There are 17 controllers on each 3-inch limb module. Due to how I2C works, a channel can have one of each controller type, but not two of the same controllers.

The selected ICs were tested together on a breadboard to verify the RP2040s are sufficiently powerful prior to finalizing the PCB design, as shown in Figure 5. However, due to time and voltage availability constraints, breadboard testing was minimal and insufficient.

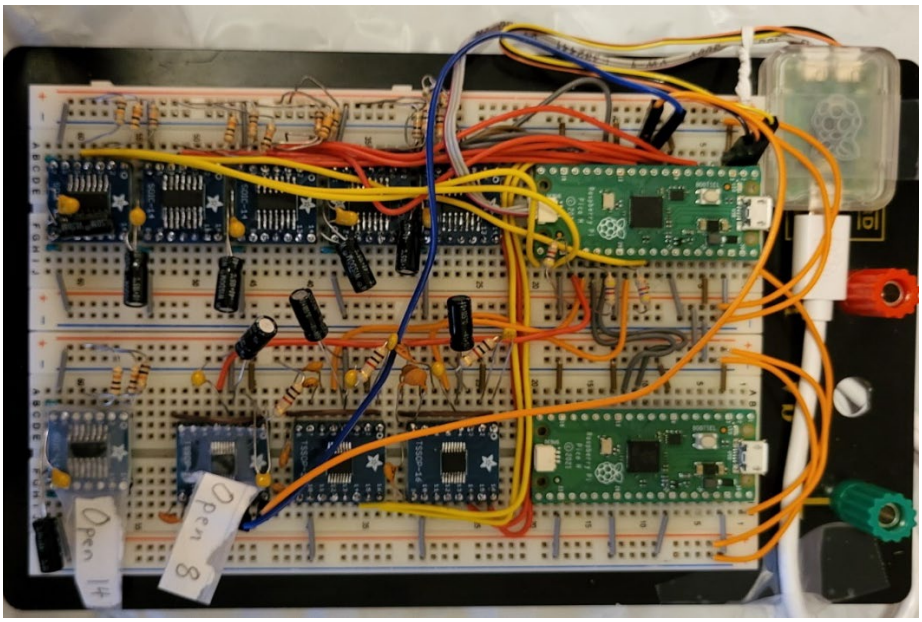

Figure 5. The breadboard used for testing prior to designing PCBs

The joint modules consist of a motor and a single Raspberry Pi Zero 2 W. These modules should contain sensors to maximize either sensor density or channel availability, but the prototype does not due to time constraints. Because joint shape requirements may vary in different arm configurations, exact specifications must be determined on a case-by-case basis. It should be noted that when this project is resumed after publication of this paper, it may be necessary to also include a Raspberry Pi Pico with each joint module for additional UART channels, because Zero 2 W only includes a single UART channel. These modules use the more powerful system because the central controller and reflex neurons will run on the joint modules.

E. Communication

There are several potential methods of communication and control which can be employed. The experimental portion of this project consists of trying out the following methods and comparing their performance. Due to time constraints, the prototype for this project only utilized a simplified form of conventional control; the remaining methods are explored as possibilities for experimentation in the future.

1) Conventional Control

The neuron layout resembles a substation or factory network, so it may be ideal to use conventional industrial protocols for controlling the system. Each neuron would be assigned an address. Neurons would send data commands directly to other neurons' addresses or multicast messages the entire network. IEC 61850 GOOSE would be ideal for multicast transmissions from sensors to reflex processors and the central controller due to its high-reliability high-speed real-time performance criteria [11]. For command messages, other industrial protocols, such as DNP3 or Modbus, may be used.

This approach has the advantage of simplified programming, as commands may be sent directly to a destination, with clarity while debugging of which sensors are reading what. It has the drawbacks of traditional networking; there are a limited number of possible addresses. More importantly, there is a limited amount of data that an individual channel can carry, limiting how many neurons the central controller can reliably monitor and respond to. While these limits are high, there are approximately 86 billion neurons in the human brain [12]. Because this technology is hoped to eventually be an option for brain-computer integration enhancements, it should ideally be possible to scale to sufficient size, and standard networks simply can't. Other issues include the potential for mis-operation due to lost packets.

2) Onion-Tree Routing

Assuming each neuron has a set number of neuron connections, a message to a particular neuron could be wrapped in instructions for the in-between neurons to pass it to a particular output connection. Before passing, each neuron would strip the instructions for itself and pass on the remaining instructions and final message. This is illustrated in Figure 6.

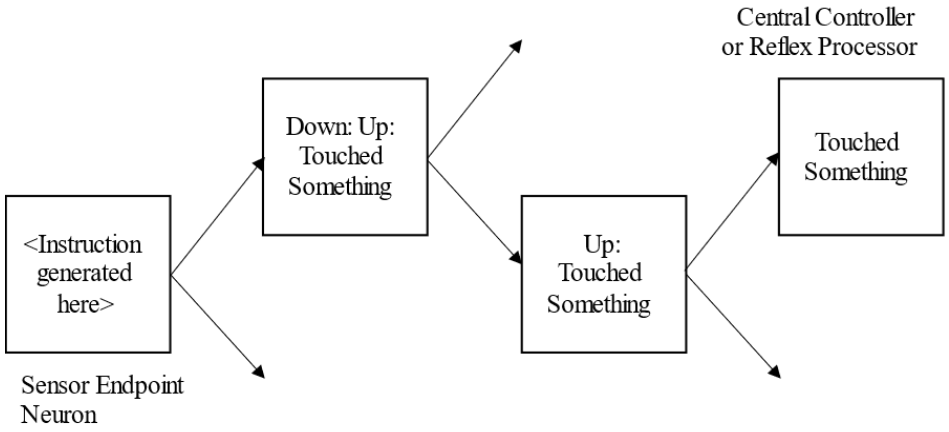

Figure 6. A very simple example of onion-tree routing between a sensor endpoint neuron and the central controller or reflex processor. Messages shown in each box indicate the message received by that box. A message tagged with "Up:" indicates that the message should be passed to the upward pointing output, and vice versa for "Down:".

This eliminates the need for limited addresses but requires a larger message to be passed the further the message's destination. Because the messages get progressively larger the further the message must go, this method limits total neurons by the bandwidth between intermediate neurons. Furthermore, it requires significant amounts of RAM, since each neuron must know the path to any neuron it might wish to communicate with.

3) Input Modification – Routing

In this control method, each output would map to a combination of inputs using a programmable "magic formula." Specialized neurons would be programmable to act upon receiving a particular set of input values. As an example, a communication neuron could divide the input value by a prime number corresponding to the output number. For example, Figure 6, the left output could

be 2, and the right output could be 3. Suppose also that the original message is 17. The sensor endpoint would send $2 \cdot 3 \cdot 2 \cdot 2 \cdot 17 = 408$, and the central controller would receive 17.

This method helps with but does not solve the increasing bandwidth issue. Either the inputs are represented as standard 64-bit binary numbers, which limits neuron distance to a maximum "maximum distance" of 63 links, or the inputs are infinite precision numbers, which will increase in size faster than simply adding a 1-bit left/right value to the front of the message.

Furthermore, this method is imprecise. If dividing would result in a decimal, the neuron could simply not send, but if two paths use the same factors in different orders, e.g. $2 \cdot 3 \cdot 2 \cdot 2$ and $2 \cdot 2 \cdot 2 \cdot 3$, multiple destinations would consider the number a valid input. This is similar to the method typical machine learning neural networks use for intermediate calculations.

4) Input Modification – Neural Network Neuron

This method, perhaps the simplest, is for the communication neurons to behave as neurons in a standard machine learning neural network (ML NN). Once again, each output would correspond to a combination of inputs via a magic formula, but the output values create functionality rather than being used for routing [13]. Although this is the simplest method to implement, it is perhaps the most difficult to program. This method also introduces uncertainty in what the arm is actually doing, as ML NNs produce their calculation weights randomly.

5) Vector Input Modification

This method is similar to the input modification method but uses vectors instead of scalars. Given that each output is assigned a different polar direction, the communication neuron can simply add the output direction to the input neuron. To prevent paths going to the same point, the input polar value could be rotated by an arbitrary transcendental angle such as Eqn. 9.

$$\frac{360^\circ}{100 \cdot \mathrm{e}} \approx 1.3244 \ldots^\circ \tag{9}$$

Equation 9 example arbitrary angle for vector input modification.

Because transcendental numbers cannot exist in computers, there would eventually be path conflicts, though they'd be rarer. This would allow for propagated floating-point error, so neurons would need an input tolerance, limiting neuron space and providing the possibility of miscommunication.

6) Integrated sub-neural networks

This idea is to simply put a machine learning neural network inside each neuron. This would be the most powerful option, but also the most difficult to program, and requires powerful module processors.

7) Biological Neuron – Chemical Neurotransmitter Simulation

This idea is likely the most difficult to program and implement, but most compatible with interfacing with human biological neurons. The idea of this method is to research and create a

simulation of a biological neuron inside each software neuron, so that the arm behaves exactly as a human arm would.

This idea is mentioned as a possibility to support the future goals of this project but is far outside the scope of this project and was not researched.

F. Control

Two primary methodologies exist for controlling the robotic arm system: manual programmed control and machine learning (ML). It is important to note that not all communication methods presented in this research are compatible with both control approaches.

(a) The programmed control approach involves explicit specification of actions for each neuron by the operator. This method follows a rule-based paradigm where software is programmed to automate specific tasks following predefined rules. A key implementation decision within this approach concerns whether the arm should autonomously override commands based on sensor data or merely report sensor information to the operator—this determination is left to the specific implementation.

(b) ML control, while potentially offering greater adaptability, requires substantial investment in terms of both time and financial resources for model training [14]. This approach presents significant challenges including the relationship between complex machine learning algorithms and limited computational resources available on robotic platforms, and the adaptation of these algorithms to dynamic, changing environments. Under an ML framework, the arm could be assigned a generalized objective rather than specific commands, relying on learned capabilities to accomplish the task.

Finally, the current architecture organizes neurons in a tree structure to facilitate prototyping, though this design decision inherently limits the system's potential capabilities. An alternative approach involving the organization of neurons as a graph structure combined with machine learning control methodologies would potentially provide enhanced functionality and flexibility; however, such an implementation would significantly increase development complexity and associated costs.

1) Confidence

Each joint controlling neuron would have a confidence value in how confident the robot is that that joint can be safely moved. Proximal joints will adjust their own confidence to be less confident than the distal joints they are attached to. A joint with low confidence will be restricted to moving slowly. A joint with high confidence will allow itself to move quickly. An example confidence algorithm is provided below after describing sensing and adaptation.

2) Sensing

Spread throughout the arm are surface sensors for detecting touch. Capacitive touch sensors detect if the arm has touched an object. Inductive sensors detect if the arm is close to touching an object but only works on metal objects. Additional sensor types may be used if needed by the

implementer. When a sensor detects something, a signal is sent back to the central controller. For capacitive sensors in positions which shouldn't touch objects, signal is also sent back to the most proximal reflex processor.

3) Adaptation

There are two conditions where the robot arm will have to adapt to new geometry: Adding a limb and removing a limb. When a limb is added, the robot no longer knows its safe degrees of freedom and needs to re-learn. To accomplish this, newly powered neurons will start at the minimum confidence level. When the overall confidence is below a threshold, the central controller operates in a learning mode, slowly flexing available joints, starting from the most distal, to determine what its available movement range is. If the arm comes close to touching an object, whether in learning mode or not, it sets its maximum range to just inside its current location. Outside learning mode, obstructions only temporarily block movement, unless they are repeatedly encountered, triggering a confidence reduction.

When a limb is removed, the movement range is not extended, so the robot doesn't need to relearn. The central controller does need to know its reduced options, so the neuron proximal to the removed neuron will send a strong "pain" signal to indicate the reduced flexibility. Response to this signal is left to the arm's programming. It may be used to alert the operator to damage, or may be ignored if the program expected a limb to be removed, for example, when changing tools.

4) Confidence example

Each module would separately track confidence as described below.

1. When powered, set confidence to a small non-zero number (operator-configurable).
2. If the module detects a possible future collision, reduce confidence by an operator-configurable value and alert proximal modules. If the detection comes from an analog sensor, the sensor's range can be used to reduce confidence.
3. If the module is detecting a collision, set confidence to zero.
4. If the arm is not detecting a collision and has passed a command distally, increase confidence up to the maximum confidence value by an operator-defined value.
5. If a distal module reports a collision, reduce confidence in the power-on value.
6. Ask the distal module(s) for its/their confidence. If it is lower than the current module's confidence, overwrite the current module's confidence with the provided value.

The operator-configured value in step 4 should be lower the more proximal the module is installed. The other values should be set by experimentally determined guidelines.

When a movement command passes through the module, its speed will be clamped to the current [confidence]:[max confidence] ratio. This means that the arm will move slowly when it detects a nearby object and comes to a dead stop when it detects a collision. When this happens, the operator should set behavior to either alert the operator to reset it, or auto-reset after an interval or if the collision stops being detected.

IV. Software Simulation

As a part of this project, a software simulation, shown in Figure 7, was created to test large networks and determine how the arm and communication methods would scale to massive size. This simulation was only partially completed. It was written in Java for rapid development, and the source code is available at https://github.com/MichaelJ-SHSU/thesis-simulation/releases/tag/thesis.

Figure 7. The simulation window while running.

The simulation uses several simplifying assumptions. Firstly, only a 2D plane is simulated. Secondly, the simulation uses a simplified form of direct control for its messaging. IEC 61850 testing was canceled upon the discovery that the specification costs CHF 23'414.-c [15], equivalent to $26 thousand dollars at the time of checking. The protocol for the simplified simulation is shown in Table 2.

Table 2. Simple packet protocol used in simulation

Field Name	Size (bytes)	Description
Source ID	8	The ID of the sending neuron.
Destination ID	8	The ID of the receiving neuron. The message may be intercepted and used by other neurons.
Message ID	16	UUID of the message
Direction	4	Direction in tree message travels. 1: proximally. 2: distally.

		3: both. Other numbers reserved.
Length	4	Length of the message
Command	4	Type of message 0: NULL – ignore this message 1: MOVE – Actuate a joint to specified value. Ignored if destination is not a joint. 10: SENSE – Sensor detected this distance. 11: SOFTSENSE – SENSE processed by reflex neuron 20: PAIN – Joint overextended or collision detected 21: SOFTPAIN – PAIN processed by reflex neuron 30: OVERRIDE_INFORM – signal from reflex neuron to central controller informing it how its commands were changed.
Data	Length-4	Function varies by command

In order to produce useful metrics for benchmarking, the simulation uses a message cycling system, where a message cannot pass between three neurons in the same cycle. The simulation outputs the messages passing through each node, joint movement, potential collisions reported by sensors, and actual collisions undetected by sensors. The end of the log for moving the joint on neuron 55 is shown in Figure 8.

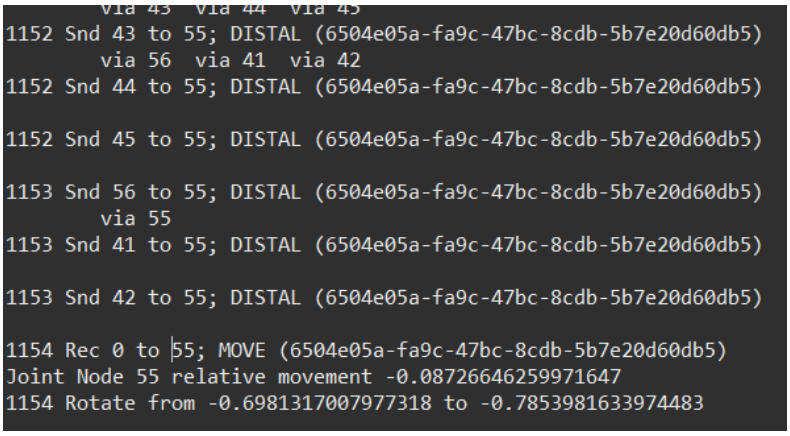

Figure 8. The arm simulation window while running.

V. EXPENDITURES

The total associated cost for the project is $1,160.59, itemized as follows:

Component	Cost (USD)
Prototyping equipment	$44.63
Five limb modules	$836.86
Two joint segments (unfinished)	$211.68

Currency conversion fees	$39.00
Manufacturing fees and unusable inventory	$28.42
Total expenditure	**$1,160.59**

Of the total project cost, $600.00 was provided by the Elliot T. Bowers Honors College as research funding, with the remaining $560.59 funded through ETEC Department.

VI. Future Goals

As this is still a work in progress project the primary objective moving forward is to fulfill all of the original project goals. This includes:

- Completion of the simulation environment with support for 2D simulation using a pre-defined benchmarking script, implementing the methods of control and communication described in § III-C.
- Programming the physical robotic arm to utilize at least one fully simulated communication method and validate its performance with real-world data.
- Comprehensive testing on both the simulation platform and physical arm to determine optimal methodologies, evaluated according to speed of command execution, reflex responsiveness to unexpected stimuli, adaptability to geometric modifications, and scalability potential. Given that the human brain contains approximately 86 billion neurons [12] and considering that this project aims to develop technology that may eventually interface with human neural systems, scalability is identified as the parameter of highest significance.

For the sake of simplicity and plausibility, the following goals were excluded from this project's scope, but may be added in the future:

- Ideally, the robot should be able to attach and remove its own limbs.
- It is desired to be able to simulate biological neurons.
- The eventual goal is that the robot's movement can be directly controlled by a person, either by directly reading the user's intent, or by mimicking one of the user's limbs.

It would be useful to integrate the neural network to a camera with an image recognition algorithm, but this is currently beyond scope.

VII. Educational Component

This senior design project was implemented as part of the ETEC 4199 – Senior Design I and ETEC 4399 – Senior Design II course sequence. ETEC 4199 constitutes a one-credit proposal defense conducted in the Fall semester, while ETEC 4399 comprises a three-credit course in which students prototype, troubleshoot, and evaluate their designs. Both courses assess multiple student learning outcomes aligned with ABET-ETAC and ATMAE standards, particularly focusing on SLO3+SLO5 (ABET-ETAC) and SLO4+SLO5 (ATMAE), which emphasize effective functioning as both member and leader in technical teams across diverse presentation formats.

The performance indicators for these learning outcomes include:

1. Use of appropriate context, conventions, and mechanics

2. Utilization of credible sources, evidence, and structure
3. Demonstration of oral presentation skills

This project represents one of eleven capstone projects completed during the 2024-2025 academic year in section ETEC 4199-01. Notably, this project was unique among the cohort as it involved a single student working independently under honors college thesis requirements and with honors college funding support. Throughout the development process, three formal in-class presentations (preliminary, midterm, and final) were conducted, providing valuable critical feedback from peer teams and faculty members. Additionally, weekly consultations with faculty advisors were held either face-to-face or via Zoom outside regular class hours to monitor progress and address design challenges.

Assessment metrics for the aforementioned student learning outcomes are presented in Table 2. Comprehensive assessment data for all student learning outcomes will be completed following the conclusion of ETEC 4399 at the end of the Spring 2025 semester, though this information will not be available for inclusion in the final draft should this paper be accepted for publication.

Table 3. Sample assessment selected for SLO3+SLO5 (ABET-ETAC) and SLO4+SLO5 (ATMAE)

Key Performance Indicators	Unsatisfactory < 60%	Developing 60-69%	Satisfactory 70-79%	Exemplary ≥ 80%
a) Use appropriate context, conventions, and mechanics (Pre-Proposal + Mid-Report)	0%	0%	0%	100%
b) Use credible sources, evidence, and structure (Final Technical Report)	0%	0%	0%	100%
c) Demonstrate oral presentation skills (Final + Mid Presentation)	3.3%	0%	0%	96.7%

REFERENCES

[1] Sudarsan, S., & Sekaran, E. C. (2012). Design and development of EMG-controlled prosthetic limb. Procedia Engineering, 38, 3547–3551.

[2] Patil, P. G., & Turner, D. A. (2008). The development of brain–machine interface neuroprosthetic devices. Neurotherapeutics, 5, 137–146.

[3] Murphy, D. P., Bai, O., Gorgey, A. S., Fox, J., Lovegreen, W. T., Burkhardt, B. W., … Fei, D. Y. (2017). Electroencephalogram-based brain–computer interface and lower-limb prosthesis control: A case study. Frontiers in Neurology, 8, 696.

[4] Neuralink. (2024, August). Prime study progress update — second participant. Retrieved January 14, 2025, from https://neuralink.com/blog/prime-study-progress-update-second-participant/

[5] Shu, T., Herrera-Arcos, G., Taylor, C. R., & Herr, H. M. (2024). Mechanoneural interfaces for bionic integration. Nature Reviews Bioengineering, 2(5), 374–391.

[6] Schultz, A. E., & Kuiken, T. A. (2011). Neural interfaces for control of upper limb prostheses: The state of the art and future possibilities. PM&R, 3(1), 55–67.

[7] Tortora, G. J., & Derrickson, B. (2012). Principles of anatomy & physiology (13th ed., p. 14). Hoboken, NJ: John Wiley & Sons.

[8] Bing, Z., Meschede, C., Röhrbein, F., Huang, K., & Knoll, A. C. (2018). A survey of robotics control based on learning-inspired spiking neural networks. Frontiers in Neurorobotics, 12, 35.

[9] Raspberry Pi. (n.d.). Buy a RP2040. Retrieved October 22, 2024, from https://www.raspberrypi.com/products/rp2040/

[10] Microchip Technology. (2015, November). CAP1206: 6-channel capacitive touch sensor (DS00001567B) [Data sheet]. Retrieved from https://www.microchip.com/en-us/product/cap1206

[11] Hou, D., & Dolezilek, D. (2010, October). IEC 61850: What it can and cannot offer to traditional protection schemes. SEL Journal of Reliable Power, 1(2). Retrieved from https://cdn.selinc.com/assets/Literature/Publications/Technical%20Papers/6335_IEC61850_DH-DD_20080912_Web.pdf

[12] Lent, R., Azevedo, F. A. C., Andrade-Moraes, C. H., & Pinto, A. V. O. (2012). How many neurons do you have? Some dogmas of quantitative neuroscience under revision. European Journal of Neuroscience, 35(1), 1–9. https://doi.org/10.1111/j.1460-9568.2011.07923.x

[13] Islam, M., Chen, G., & Jin, S. (2019). An overview of neural network. American Journal of Neural Networks and Applications, 5(1), 7–11. https://doi.org/10.11648/j.ajnna.20190501.12

[14] Wang, S., Zheng, H., Wen, X., & Shang, F. (2024). Distributed high-performance computing methods for accelerating deep learning training. Journal of Knowledge Learning and Science Technology, 3(3), 108–127. https://doi.org/10.60087/jklst.v3.n3.p108-126

[15] International Electrotechnical Commission. (2025). IEC 61850:2025 SER. Retrieved from https://webstore.iec.ch/en/publication/6028

APPENDIX

The hardware organization diagrams for the modules used by the prototype for this project are provided in Figure A1, A2, and A3.

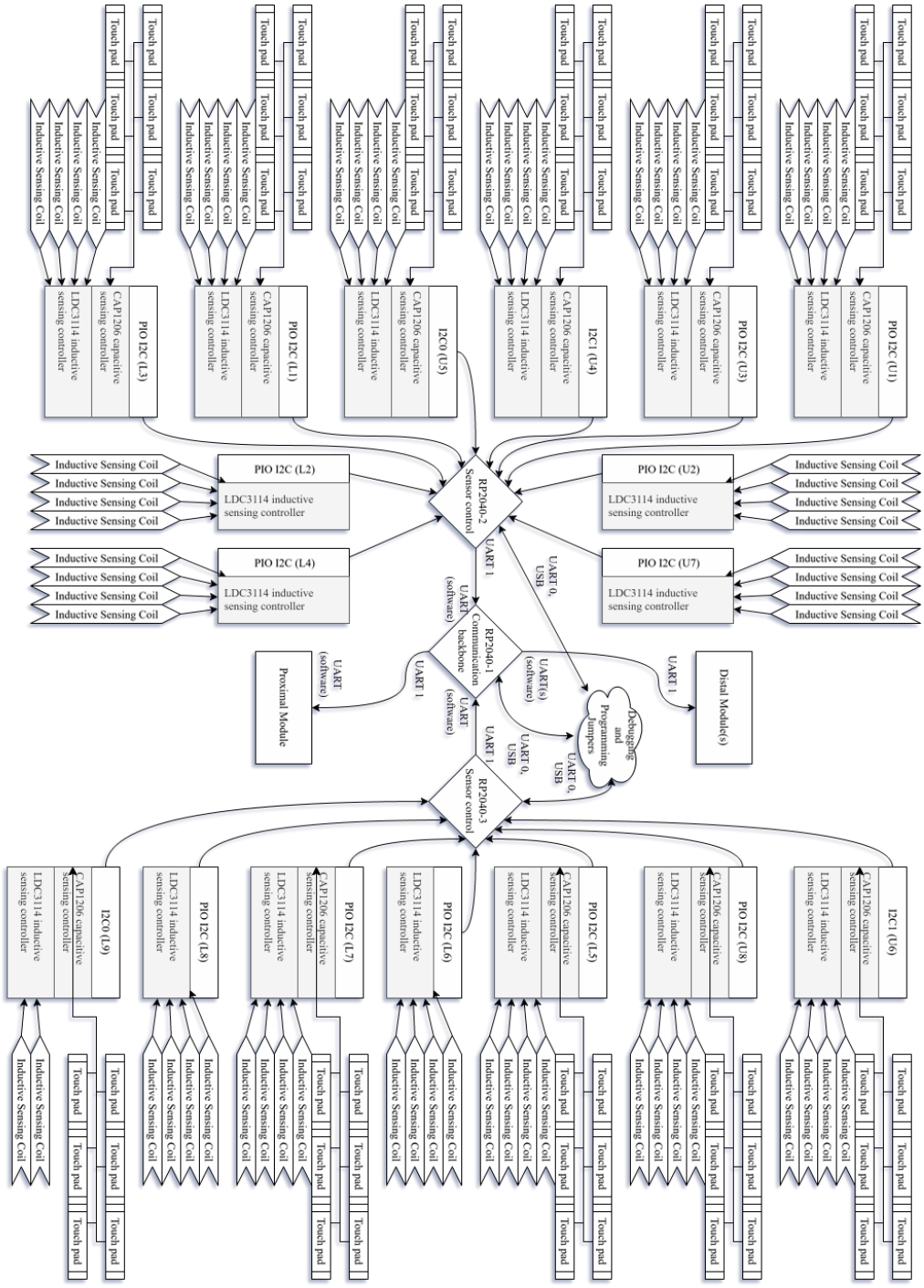

Figure A1. The organization of the sensor modules in this project. Designed for 3 inches of a limb with a 3-inch diameter. Implementers need not follow this organization; modify it to suit the needs of the limb being designed.

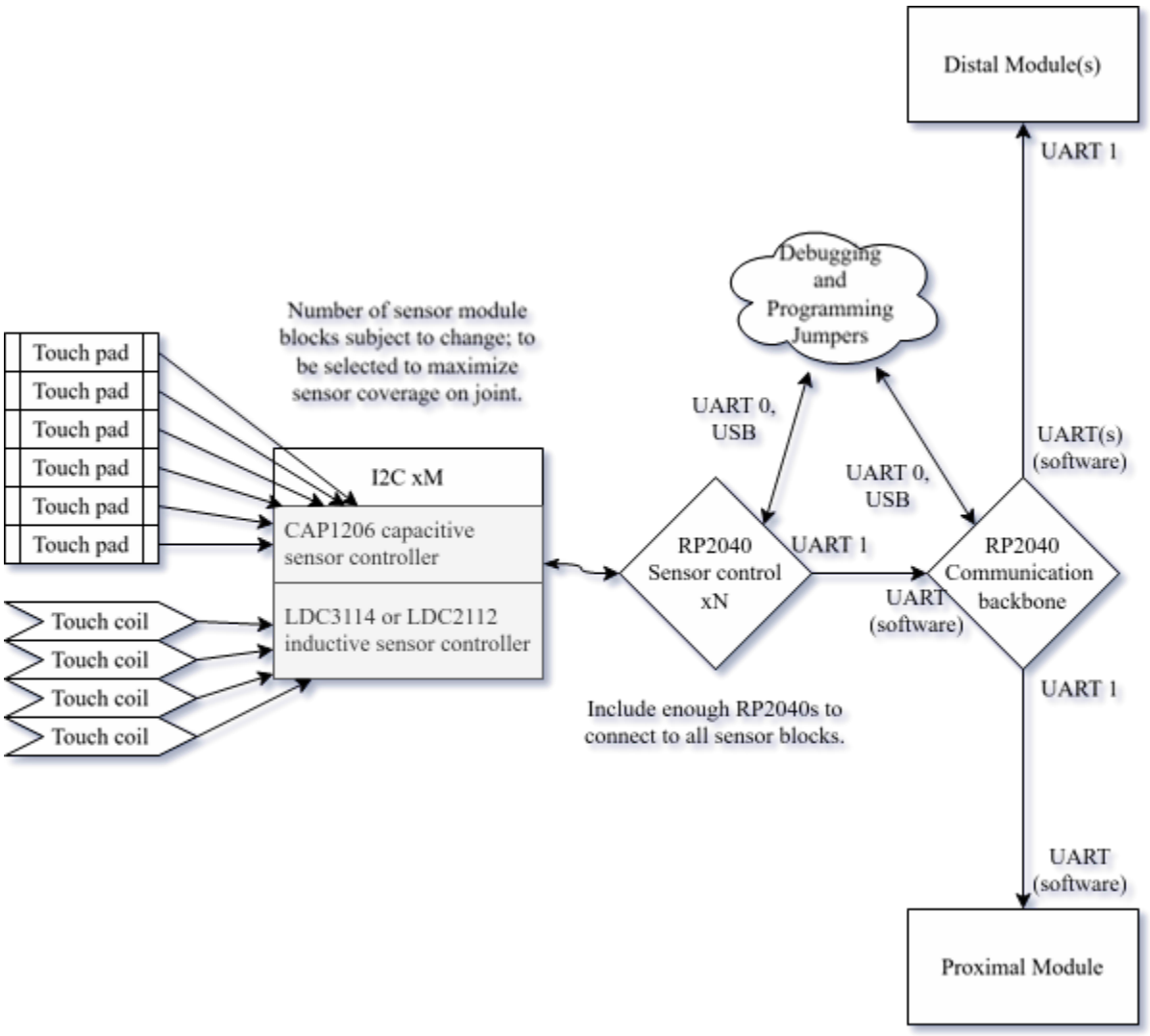

Figure A2. Generic sensor module block diagram.

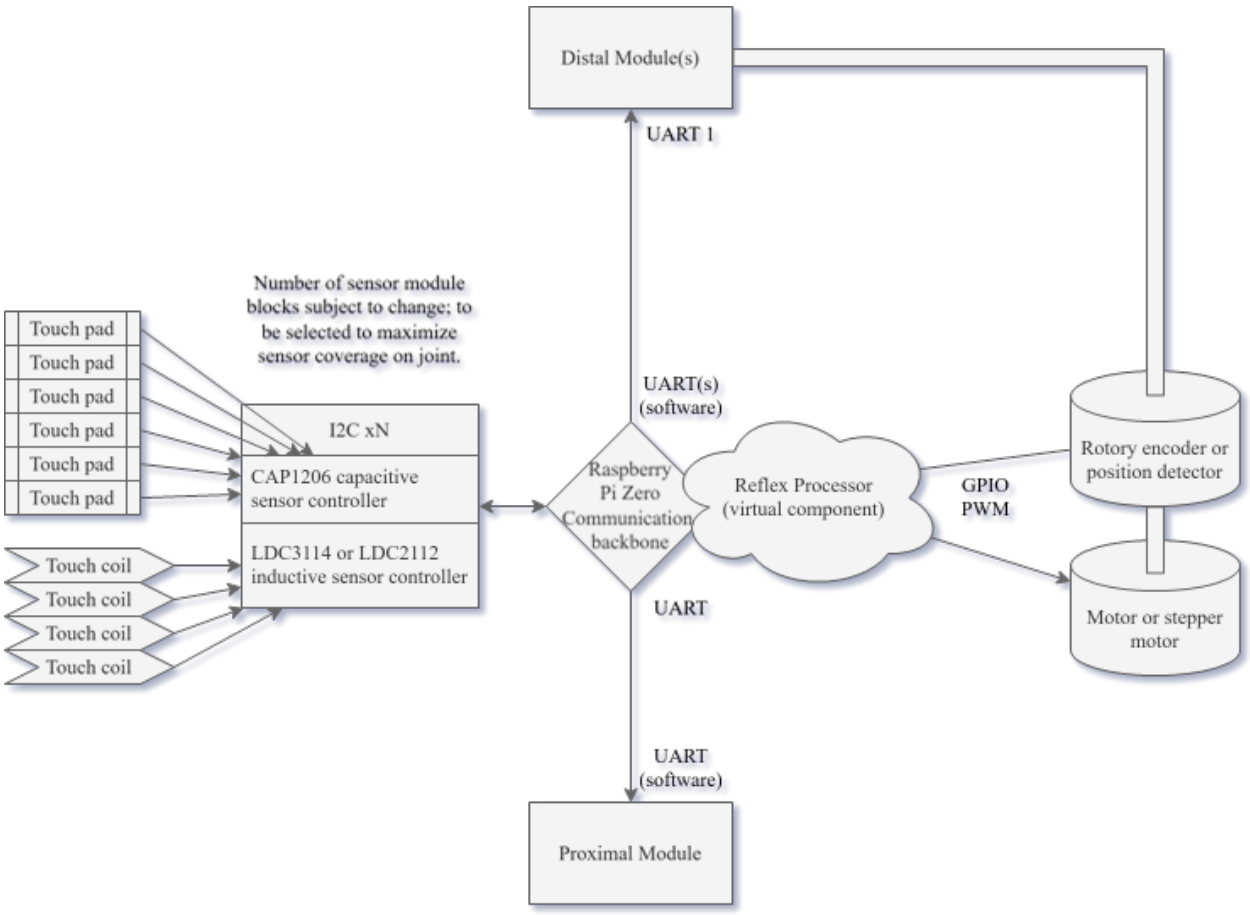

Figure A3. Generic joint module block diagram.