HUEBERT Design Document

Matthew Ebert

Jamieson Fregeau

ECE 356, UVic

2021-11-18

Table of Contents

Introduction

The following document contains the implementation of the HUEBERT design. While the design is in a functional form after the first sprint of development, several functionalities, safety, and security measure still need to be implemented. A note will be made in areas where the product is not in its final form.

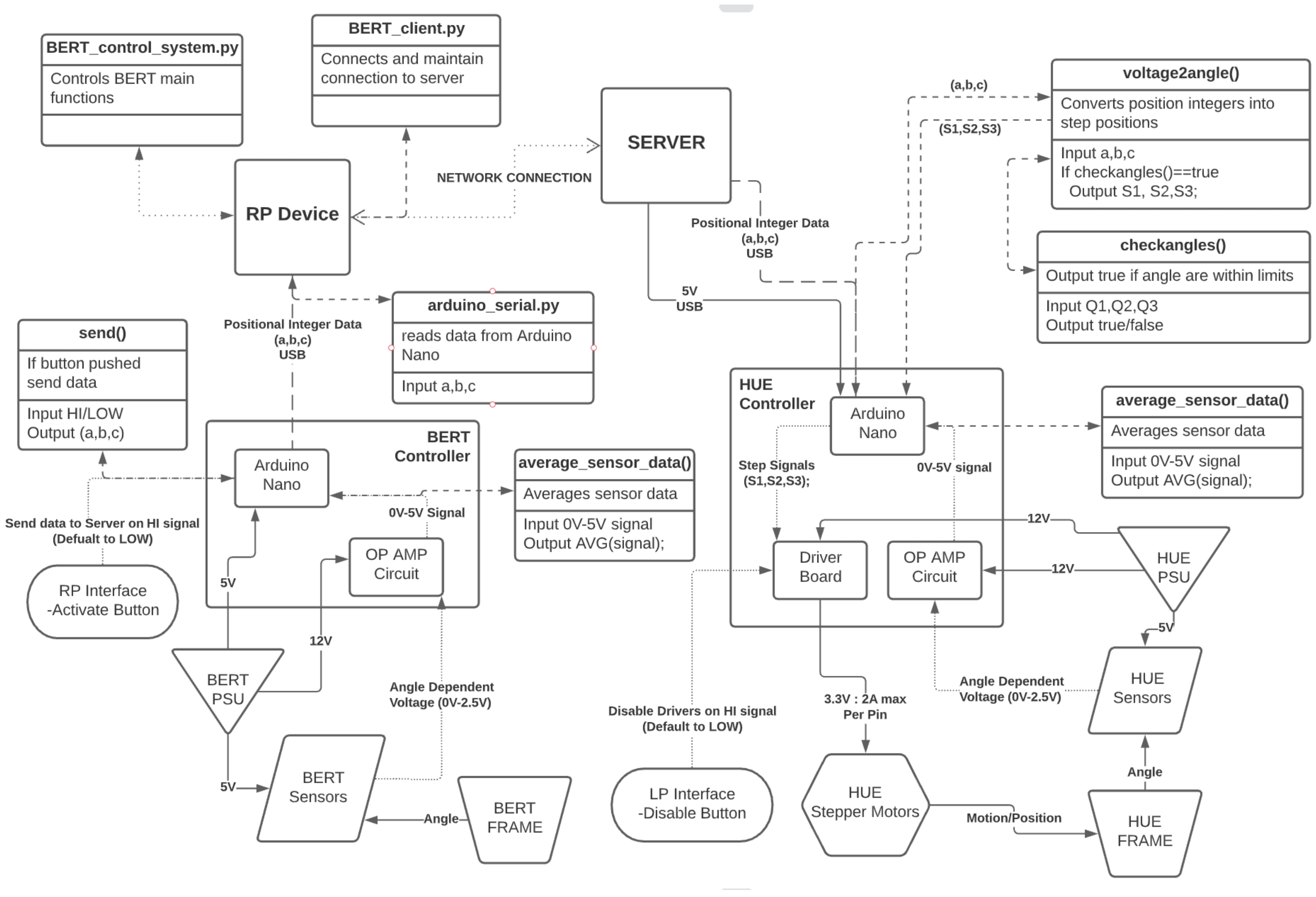
1. HARDWARE/SOFTWARE COMPONENTS

The following diagram illustrates the interaction between hardware components and corresponding software in the current version of the HUEBERT system. Note this version is not in its final form and only demonstrates the implementation up to date. A more abstract and complete diagram is detailed in the following section based on design diagrams. Use the following interpretation rules to read the diagram below

- Arrows indicate connection. A common beginning or end point of multiple arrows demonstrate a transfer relationship
- Square boxes are controllers (Microprocessors, Logic, Circuitry)
- Boxes labelled with names ending in () or .py indicate software functions
- Small, dotted lines indicate signals
- Dashed line indicates software input output
- Large, dashed lines indicate serial connection
- Solid lines indicate power or mechanical connection

1.1 Implementation Diagram

This diagram represents HUEBERT after the first sprint of development.



1.2 Original Design Diagram

For reference, the original design diagram is included below for a more abstract representation of the system.

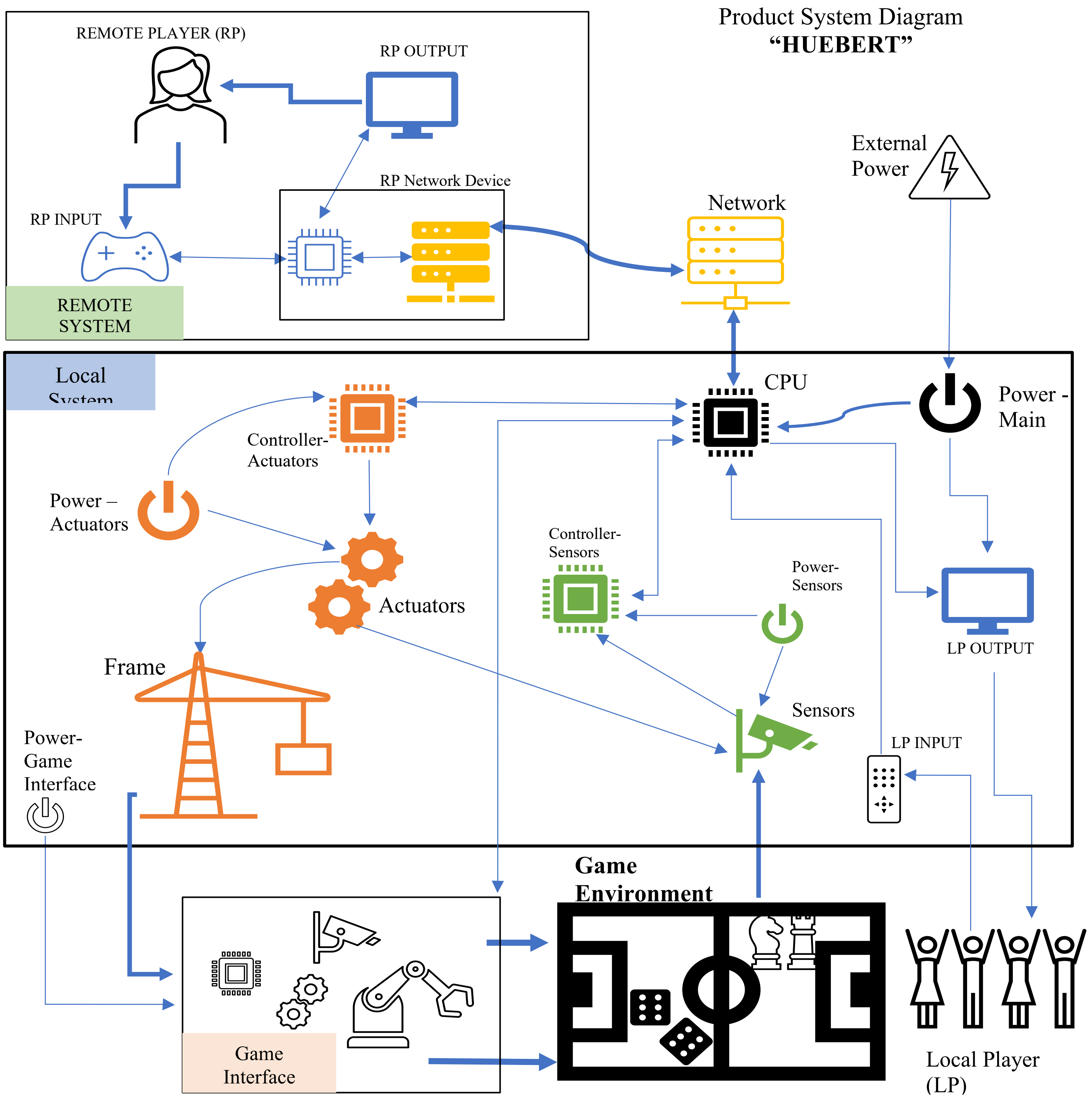


Figure 1: HUEBERT system diagram

2. Detailed Specifications

2.1 HUE

HUE, the local system for interacting with the local players (LP) and game environment, consists of a 3D printed frame, positional sensors, stepper motors, LP interface, a power supply, and a controller unit. A description of all these components and subcomponents can be found below.

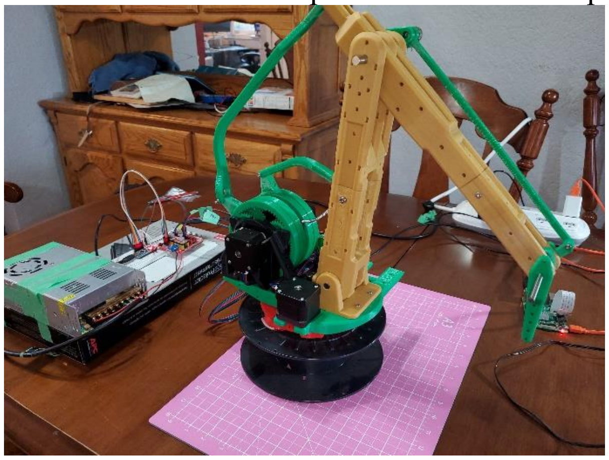


Figure 2: The HUE system

2.1.1 HUE Controller

The HUE controller consists of an Arduino Nano, a motor driver board and drivers, and an RC op amp circuit.

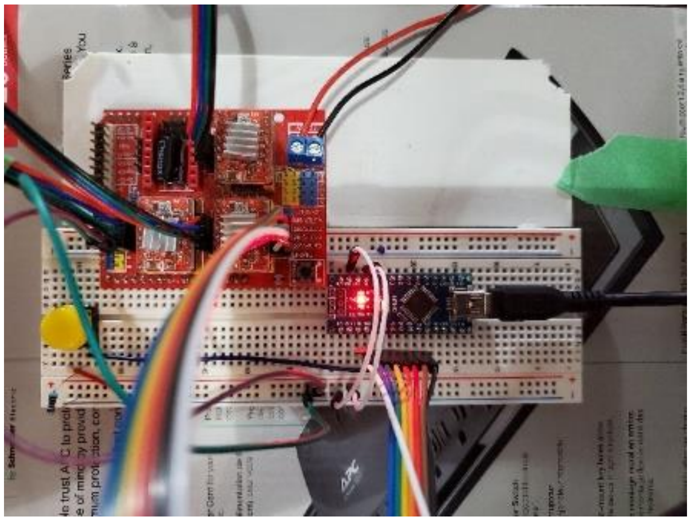


Figure 3: HUE controller: RC op amp circuit (not in photo)

The Nano controls all of HUE's processes and handles all incoming and outgoing transmissions to the Server.

The Nano is connected to several interfaces including:

- Stepper Motor Drivers
- Positional Sensor Circuit (RC op amp circuit)
- Raspberry Pi (Server)
- LP Input
- Power

The Nano handles these interfaces through analog and digital pins. The Nano receives power from the USB connection to the raspberry pi. Since the current requirements of the Nano are small, only a common ground needs to be connected to HUE's power supply for sensor reading purposes.

The operational amplifiers and RC circuits serve to filter and amplify the potentiometer sensor voltage returning from the positional sensors. The incoming voltage is filtered for AC irregularities before being amplified from a range of 0V to 2.5V to 0V to 5V (Gain =2). This allows the analog pins on the Nano to use their full range of precision (0 to 1024). The operational amplifier receives 12V from HUE's power supply.

The A4988 motor drivers convert the digital pin output from the Arduino Nano into stepping instruction for HUE's stepper motors. The motor drivers are powered from 2 sources: 12V from the power supply is supplied to the driver board and 5V from the Nano supply's the voltage to the logic functions of the board. Several capacitors filter voltage from the power supply and stabilize the DC input to the stepper motors. 3 motor drivers are used (1 for each motor) and compiled together on a single board. Each driver has an individually set current limiter which is set based on the torque requirements of each motor.

2.1.2 HUE Frame

HUE's frame can be considered a robotic arm consisting of a 4-bar and 5-bar linkages. The driving points to control the motion of the frame are located on the pivot bearing assembly and consist of 2 4:1 herring bone planetary gear sets (for controlling the linkages) and a 13:1 spur gear (for controlling the z rotation). HUE's mechanical frame consists of mostly 3D printed components. These components function as the:

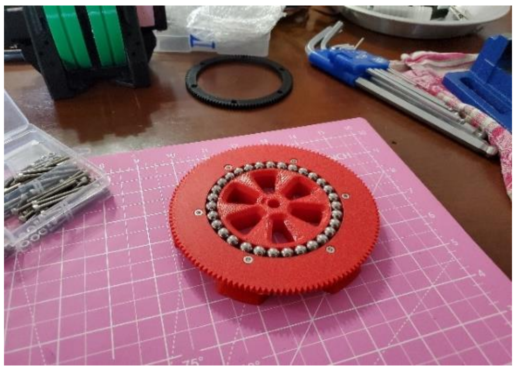
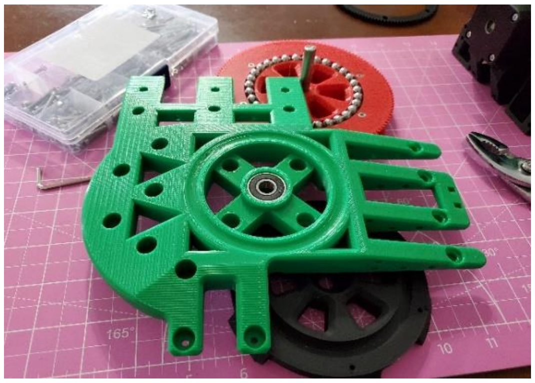
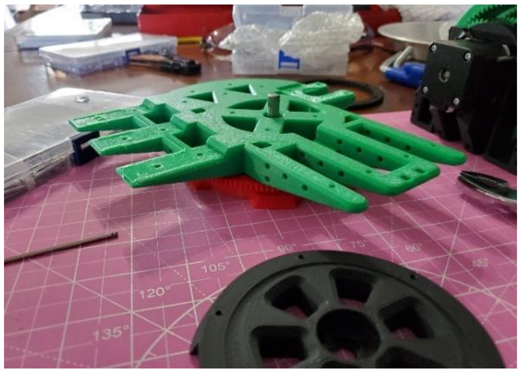
- arm main links
- control links
- herring bone planetary gears
- rotating base
- motor mounts
- bearing seats

The non-printed components consist of:

- 8mm and 5mm sealed bearings
- 8mm and 5mm axles
- M4, M3, and M2 fasters (bolts and lock nuts).

- 8mm ball bearings
- 20 kilo Ohm potentiometers.

The images below summarize the assembly of the frame.

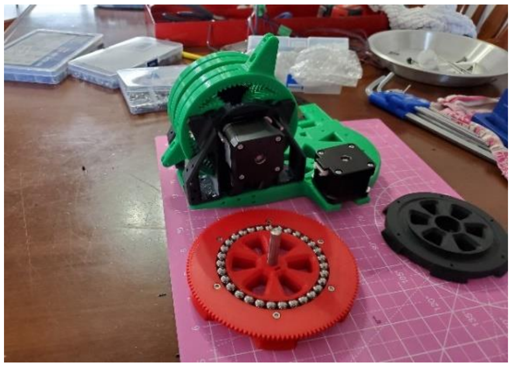
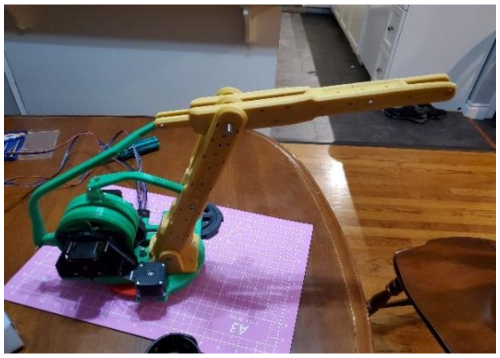
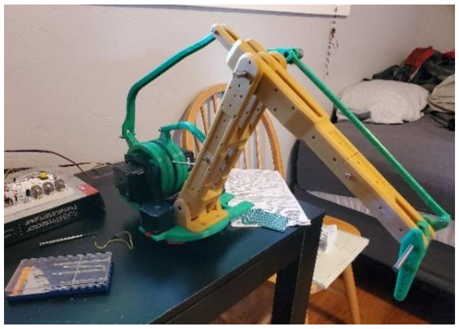
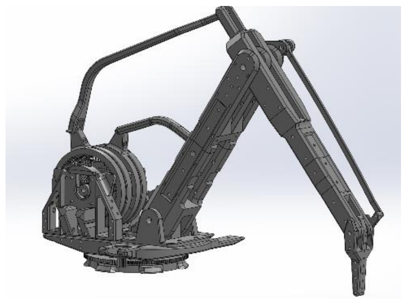





Figure 4: HUE's Frame Assembly and model

2.1.3 Positional Sensors

HUE’s positional sensors consist of 2 potentiometers. These variable resistors give an angle relative voltage from which HUE’s controller can decipher the current position of HUE’s arm linkages. These sensors are powered by 5V from a voltage divider supplied from HUE’s power supply.

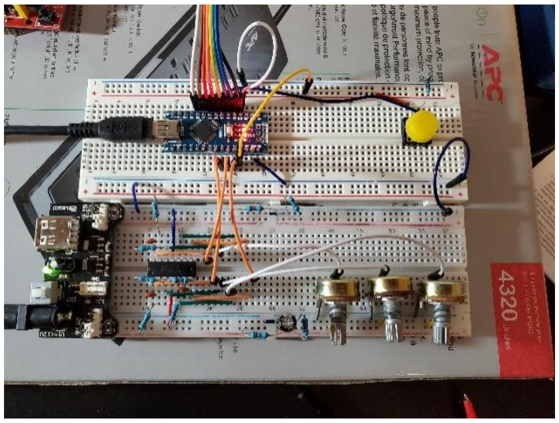


Figure 5: Potentiometers mounted on prototyping circuit used for testing accuracy of the sensors.

2.1.4 Stepper Motors

HUE’s actuators consist of 3 Nema 17 stepper motors. These motors operate at 3.3V with a maximum current of 2 amps. They are mounted on HUE’s frame at the 3 driving points and connect through 4 wire cables to the pins of the motor driver board where they are controlled by the A4988 drivers.

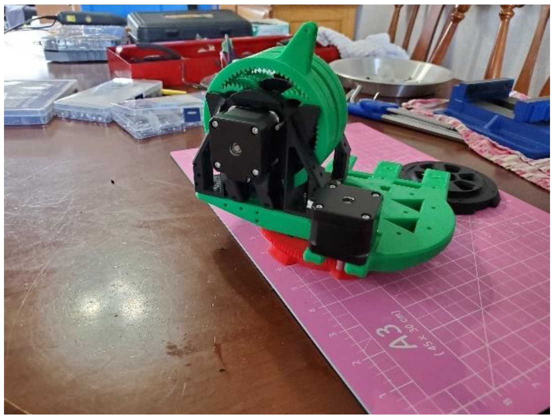


Figure 6: Nema 17 Stepper motors mounted on a portion of HUE's frame

2.1.5 LP Interface

Currently, the LP interface consists of a single button used for disabling HUE’s mechanical motion. The push button, when held, sends a 5V signal to the motor drivers with directly disables the stepper motors power. When the button is released, the motors regain power and will move as instructed. This system is independent of the Arduino Nano and will function regardless of the software state.

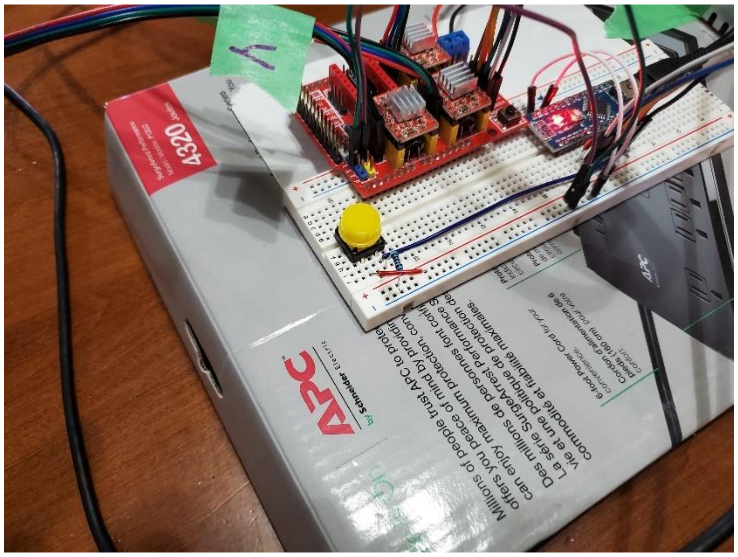


Figure 7: The disable button of LP interface (Yellow button);

This interface will be developed further in future sprints.

2.1.6 Power Supply

The power supply consists of a large 120V AC to 12V DC power supply. The component has multiple leads that deliver a steady 12V signal to the components of HUE. The supply has fuses to counteract current surges or shorts in the system.

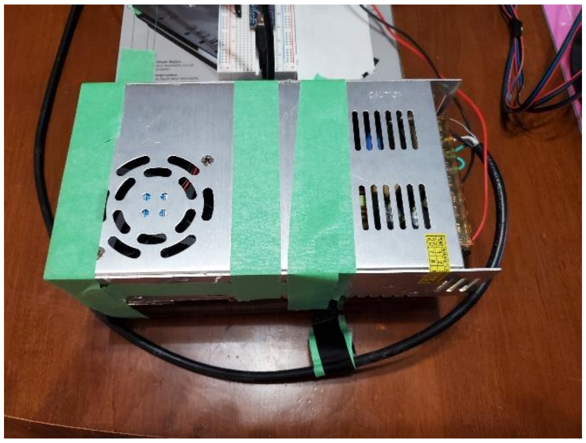


Figure 8: HUE's Power Supply (PSU)

2.2 BERT

BERT, the remote system used by the remote player (RP) for controlling HUE, consists of a controller, positional sensors, a moveable 3D printed frame, RP interface, and a power supply.

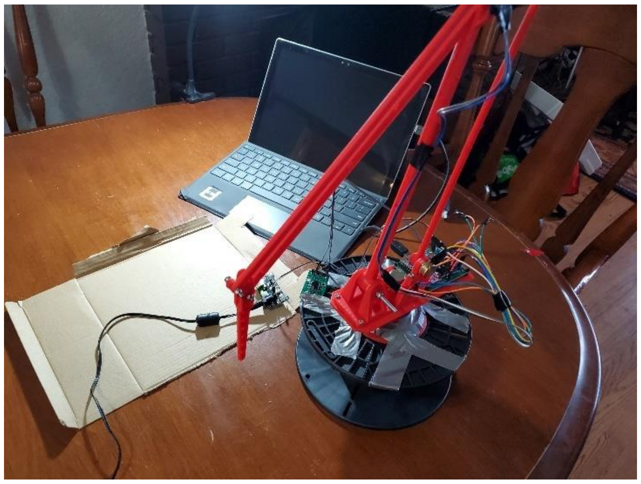


Figure 9: The BERT system

2.2.1 BERT Controller

BERT controller consists of an Arduino Nano, RC circuits, and operational amplifiers.

The Arduino Nano handles all communication with the server and user input data received from the positional sensors and RP interfaces. The Nano is supplied 5V from BERT's PSU through a voltage divider circuit.

The operational amplifiers and RC circuits serve to filter and amplify the potentiometer sensor voltage returning from the positional sensors. The incoming voltage is filtered for AC irregularities before being amplified from a range of 0V to 2.5V to 0V to 5V. This allows the analog pins on the Nano to use their full range of precision. The operational amplifier receives 12V input from BERT's power supply.

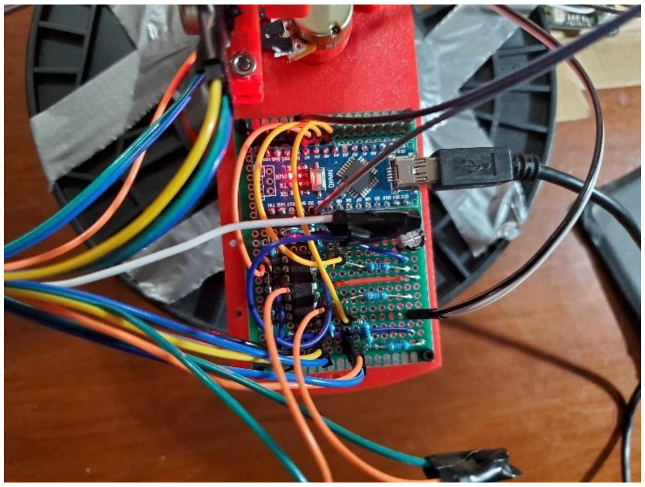


Figure 10: BERT controller

2.2.2 Positional Sensors

BERT's positional sensors consist of 3 potentiometers. These variable resistors give an angle relative voltage from which the RP can indicate to BERT's controller what input to send to

BERT. These sensors are powered by 5V from a voltage divider supplied from HUE's power supply.

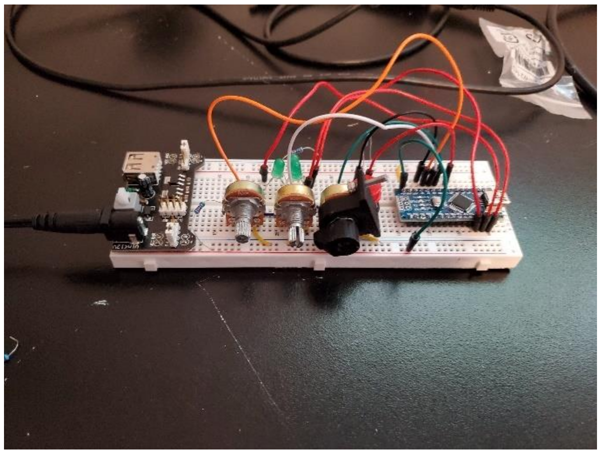


Figure 11: Prototyping board with potentiometers

2.2.3 BERT Frame

BERT Frame consists of mostly 3D printed parts. The OTS components consist of:

- M4, M3, and M2 fasteners and lock nuts
- Potentiometers
- 8mm bearing and shaft

The frame can be moved easily by the RP. The angle of the arms and rotation can be determined by the potentiometers. This characteristic allows the RP to control HUE by directly moving BERT in the way they wish to move HUE.

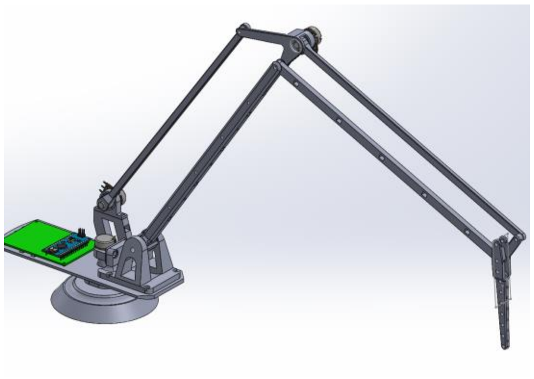


Figure 12: Model of BERT Frame

2.2.4 BERT PSU

BERT receives power from a modified OTS power supply unit which converts 120V AC to 12V DC. Voltage divider circuits are used to supply 12V to the operation amplifiers and 5V to the potentiometer and Arduino Nano.

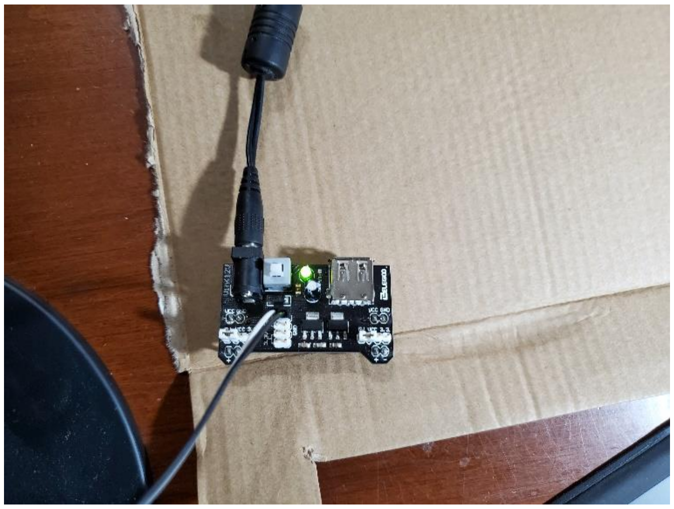


Figure 13: BERT's PSU

2.2.5 RP Interface

Currently, the RP interface consist of an activation button which sends a HI or LOW signal to the Arduino Nano in BERT's controller. The Nano uses this signal to determine when to send data to the RP device.

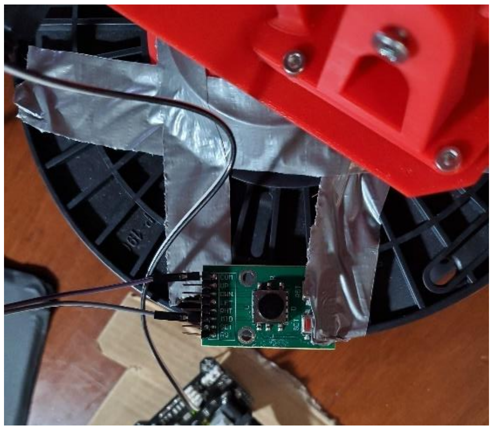


Figure 14: RP interface (Button)

This interface will be developed further in future sprints

2.2.6 RP Device

The RP device consists of a laptop or a similarly networked device on which video and audio can be displayed. It must be able to run BERT's executable files and connect to the server.

2.3 SERVER

The server establishes and secures the connection between HUE and BERT and enables the transfer of positional control data from the RP to the LP. The server is hosted locally in the HUE system on a raspberry pi and utilizes a client-server socket relationship with BERT to achieve secure data transfer.

2.3.1 Socket Client

The socket client is the BERT system and is currently ran on the laptop that accompanies BERT. The client establishes a connection to the server through a specified IP address and port and reads positional serial control data in from BERT through BERT's Arduino Nano. The client verifies the correct format of the positional control data and sends it to the server in 13-byte packets.

As this is the first sprint, there are still implementation limitations. Currently, HUE's live video stream is only displayed locally. In the next sprint, the client's functionality will be updated to include receiving the HUE's live video stream over the server and displaying it locally for the RP to view.

2.3.2 Socket Server

As discussed previously, the socket server is hosted on the raspberry pi situated in the HUE system. The server opens a socket connection on a specified port for a specified IP address and reads positional control data in from the client. The server verifies the format structure of the data and passes it to the HUE system.

The raspberry pi server is also responsible for the local video stream. This video stream originates from a raspberry pi camera and is currently streamed locally for HUE to a monitor. As discussed above, in future sprints, the server-client socket will be updated to securely transmit this video feed.

3. Software

HUEBERT's software is included in the attached compressed folder entitled IMPLEMENTATION_CODE.

3.1 BERT_control_system.py

BERT_control_system.py is the program that runs on BERT's accompanying pc and controls BERT's main functionality.

Once this program is activated, it attempts to connect to the attempts to establish a connection to the server through a specified IP address and port. Once this has been done, it continually loops through receiving positional control data from BERT, verifying it, and passing it on to the server. Should the client not be able to initially establish a connection to the server, after ten retry attempts a dedicated fatal error will occur.

In future sprints, functionality for a user to be prompted for a secure password to connect to the server is intended to be implemented.

3.2 BERT_client.py

The file BERT_client.py contains a class named BertClient to communicate with the raspberry pi server. The class BertClient contains six methods which are utilized from BERT_control_system to interact with the server.

- **__init__():** Python's built-in class constructor. Set to initialize our object with an IP address, port, number of connections retry attempts and the default socket object.
- **connect():** A method to connect to the server. Attempts to do so retry_attempts number of times.
- **disconnect():** Disconnect from server
- **send_data_to_server():** Sends control data in bytes to server
- **receive_data_from_server():** Receive control data from server in bytes
- **print_server_data():** Decode and print data incoming from the server

3.3 arduino_serial.py

This file contains a class named ArduinoPy built to establish serial communication between BERT's control Arduino and BERT's local pc. This class has three methods which are utilized from BERT_control_system to establish serial communication.

- **__init__():** Python's built-in class constructor. Set to initialize our object with a port, baud rate and timeout to utilize for communication and the default serial object.
- **read_control_data_from_bert():** Reads serial data from BERT's control Arduino. Only returns the data if it can be locally verified as the correct format.
- **control_data_verified():** Verifies the format of the incoming control data from BERT. Checks the length of the serial data and that it includes an initial positional control character such as *p* for position or *a* for acceleration.

3.4 BERT_embedded.ino

This is what controls BERT's control Arduino's functionality. This program has 4 main tasks.

1. Read analog input from BERT's potentiometers at analog ports 5, 6 and 7.
2. Average these potentiometer values over an interval, currently set to 15 milliseconds. This averaging has been put in place to stabilize the readings from the potentiometers which by default were found to vary in their precision.
3. Format the potentiometer readings into a 13-character code. This code includes the single control character *p* to signal positional data, plus the 9 potentiometer values and finally the delaminating character.
4. Send the 13-character code as serial data to BERT's accompanying pc and process it in BERT_control_system.py.

3.5 HUE_embedded.ino

This code initiates and runs the HUE controller. The code has 4 main tasks:

1. Serial Communication: The code receives data from the server and send diagnostic and sensor data to the server.
2. Voltage to Step conversion: Positional data sent by the server is converted to stepper position data for HUE's stepper motors. During this task, the validity of the motion is checked with known maximum and minimum position values.
3. Step Motors: The code sets target locations for each stepper motor and calls for a step when necessary.
4. Receive Sensor data: The code averaged potentiometer values over a time for comparison with position data. These sensor values are also used to indicate if HUE is at the limits of its mechanical motion.

Serial communication is set as a priority. At most points during execution, receiving new serial data will send the program to a read serial state.

If the received position data from the server is not valid, no motion will occur.

If new serial position data is received while HUE is in motion, the new data supersede the old data and HUE will move to the new desired position.

While no data is being received and motion has stopped, HUE will maintain its current position and indicate to the server it is ready for new information.

4. Software Industry Standards

Software industry standards have been followed throughout the development of HUEBERT in the following manners.

4.1 Code Formatting

HUEBERT's functional code has been held to a high standard of formatting and styling. For the python code, the formatting was derived from PEP8, which is the python industry standard for formatting. For the C++ code, it was formatted in accordance with the Arduino standards.

Ensuring HUEBERT's code follows a standardized formatting ensures that any area of the codebase is easy to understand for someone who is unfamiliar with it. Furthermore, it ensures HUEBERT is easy to maintain and quick to debug when the code is commented and styled in a standardized manner across the repository.

4.2 Linter

A linter was used throughout the development of HUEBERT's codebase. For python, the *flak8* linter was utilized while in C++ the *cppcheck* linter was utilized. These linters not only provide extra checks for formatting and styling standardization, but also perform statics checks for any uncaught bugs or security issues.

4.3 Code Review

Throughout the development of HUEBERT's codebase, industry standard pull requests and code reviews were performed to ensure both members of the team had the opportunity to review each other's code. This industry standard practice ensured HUEBERT's codebase was held to a higher standard and that there were no missed bugs.

4.4 Test Driven Development

HUEBERT's codebase was developed through a test-driven development framework. Before any code was written, all the test cases HEUBERT was required to pass after the first sprint were decided upon. This industry standard practice ensured all members of the team agreed upon the final functionality of HUEBERT and were both working as efficiently as possible towards this goal.

5. Hardware Industry Standards

Hardware industry standards have been followed in the following manners.

5.1 Power Supplies

All used power supplies connect to 120V outlets. These connection use standard 3 prong wall outlets which are securely insulated and covered. The AC to DC transformers is current limited to prevent excessive power flowing through any connected circuit. All capacitors in the conversion circuits have a sufficient capacitance as to not explode under normal use.

The HUE power supply, which draws current for the stepper motors, has a fuse to prevent damage to circuitry in the event of a short.

BERT's power supply is limited to a very small current so that excessive power does not flow through circuitry.

5.2 Wiring and Soldering

Consistent color coding was implemented for all wiring in the HUEBERT electrical systems. Ground, HI voltage, and signal wiring have distinct colors.

Circuits were soldered on standard prototype PCB boards. All soldered points were securely connected and checked for shorts with an ohm meter. Standard solder and soldering techniques were implemented for all circuits.

5.3 Current and Voltage Regulation

All stepper motors were connected to a power source through an appropriate and standard driver board. These board has current limiting capacities which were tuned in compliance to the Nema 17 stepper motor specification.

High ohm resistor and capacitors were used for all voltage divider circuits to prevent over heating or destruction of components under heavy load.

5.4 Kill Switch and Power Off

All electrical circuits and hardware have easily accessible off or kill switches. These are insulated and apart from high current circuits to prevent possible electrocution.

5.5 Material and Component Strength and Weight

The 3D printed components were printed with standard infill percentages and appropriate materials.

All components were printed from PLA for health considerations. The bearing capacity of PLA was confirmed to be sufficient for this application. Load bearing components were printed at an infill density of 35% for a maximum strength the weight ratio. All non-load bearing components were printed at 20% infill to reduce weight and torque requirements of the stepper motors.,

6. Safety and Privacy

Safety and privacy as determined to be paramount at the beginning of the development of HUEBERT.

6.1 Electrical Hazards

The entire HUEBERT system's electrical wiring has been implemented with a focus on safety. As HUEBERT's electrical system is quite extensive, as well as one of the key areas where safety is extra important, the following guidelines were followed when implementing the electrical system.

- All exposed wiring connections have been insulated, covered, and strapped down. This insulation and covering of these wires help mitigate the hazards posed by someone interacting with a system that has current running through it. Strapping the wires down mitigates the hazard of wires being caught in HUEBERT's mechanical motion and pulling loose and/or causing a short circuit.
- All power supplies have been current limited to the rated current for the components they are powering. This way there is not excessive current flowing through HUEBERT, and the system is operating at maximum efficiency.
- All power supplies have had appropriately sized fuses wired in series with their output to ensure no current spikes can cause damage to HUEBERT or harm to anyone nearby.
- All potentially hazardous wiring connections were soldered onto a bus board to mitigate any short circuits in the system.
- All wiring followed proper industry standards for color coordination. This guideline ensures that all ground, signal, and HI voltage wires have distinct color schemes and will not be confused for different wire.

6.2 Physical Hazards

Mitigating potential physical hazards caused by the HUEBERT system has been an important step of the implementation. As such, the following guidelines have been implemented to ensure that HUEBERT is not a physical hazard.

- HUE has been programmed to move at a speed deemed non-dangerous to nearby LPs. When HUE is in movement, there is little to no risk of an LP being caught off guard by unexpected movement that is too quick to avoid.
- HUE's control servos operate at two distinct sound levels. The first level is when HUE is stationary and not attempting to move, this is quiet but can still be heard withing 1 meter of HUE. The second sound level occurs when HUE begins to move to a new position. The servos operate at a sound level that is loud enough to easily hear within 1 meter and as such it gives any LP a clear indication that HUE is preparing to move.
- HUE has been programmed with a local override button that immediately stops all movement of HUE. This button is the first point of contact for any situation where HUE is not acting as expected and is situated in a clear location where the LPs can easily shut HUE down should something go awry.
- BERT has been programmed with a local button that controls the sending of positional data. If this button is not pressed, then the BERT system does not send control data to HUE. This way, if something is working in an unexpected manner at HUE, and the RP

sees this through the connected video stream, they can immediately stop sending HUE control data.
- Both HUE and BERT have been designed and implemented in a lightweight manner that does not pose many physical hazards. Both are light enough that they would not likely cause harm to an individual should they be hit with a moving arm, or if either system was dropped from 1 meter onto them.

6.3 Security

Security and privacy have been very important throughout the development process. As such, the client-server socket connection was decided upon for the best mode of communication from a security standpoint as it protects the privacy of its users quite well in comparison to other implementation techniques.

However, as security is such a priority for the HUEBERT system, there are always improvements to be made in future sprints. The following two server-client security updates will be made in later HUEBERT iterations.

- **OpenSSL:** The server-client socket will be encrypted utilizing the OpenSSL python library. This library enables the use of the Secure Sockets Layer – which is the same form of encryption utilized by HTTPS. This would encrypt the connection between the server and client and dramatically increase the difficulty of a malicious user to penetrate the network.
- **Password Protection:** For the client to connect to the server, a randomly generated password which is only available to the server will have to be entered. This will ensure that no third parties can connect to the server without authorized use.

7. Product Working

The following sections show the working sequences of the HUEBERT system. Detailed specification for each component can be found in the Section 2.

7.1 System

The figure below shows the BERT system. It includes BERT’s Frame, controller, PSU, RP interface, and the RP network device and display.

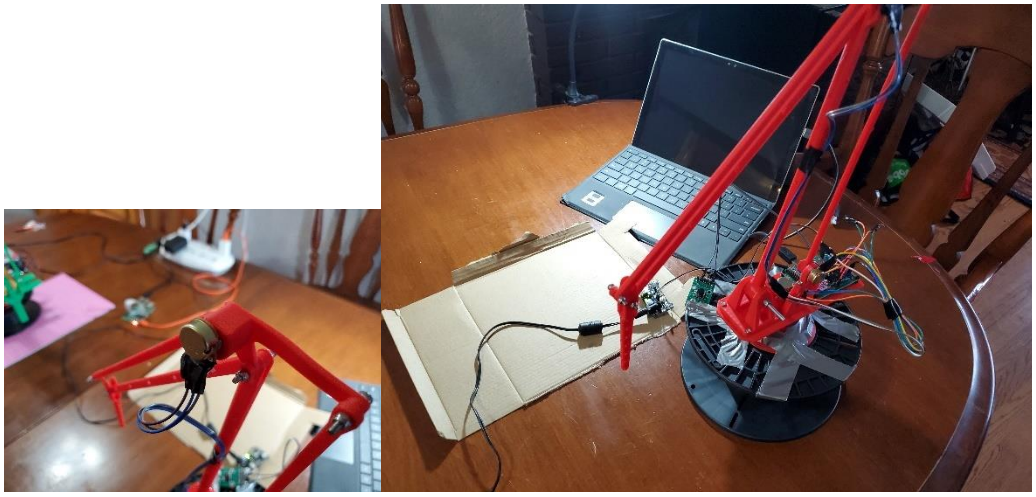


Figure 15: The BERT system

The figure below shows the HUE system. It includes HUE's Frame controller, actuators, PSU, LP interface.

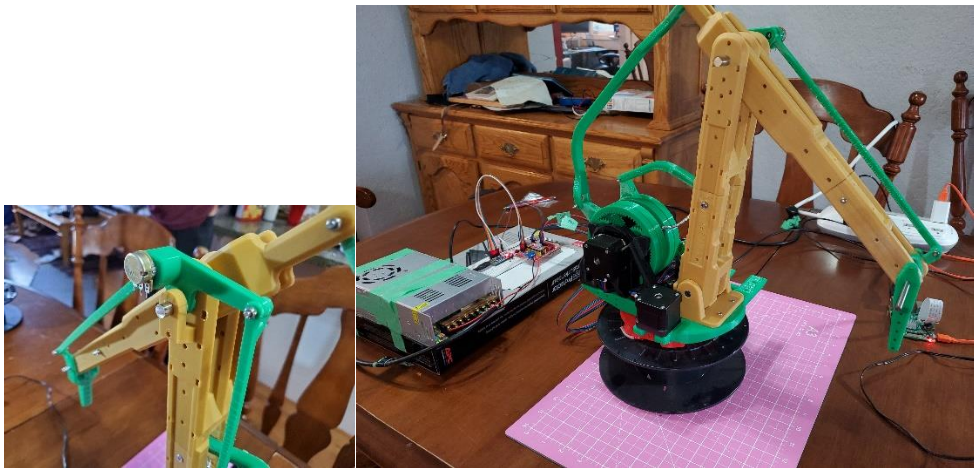


Figure 16: The HUE system

7.2 Walk Through

This section details a walkthrough of the startup and use of the HUEBERT system.

To begin, BERT's power supply is switched on, supplying 12 V to BERT's controller and sensors.

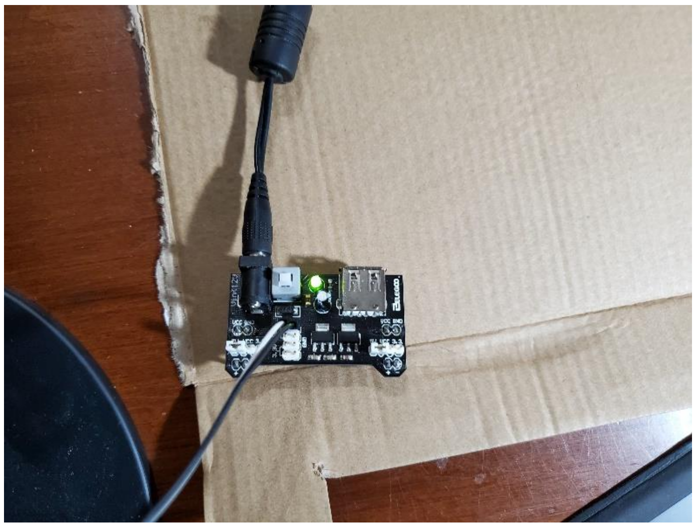


Figure 17: BERT power supply

Next, the Arduino Nano in BERT's controller is powered on and runs set up code. It establishes a serial connection with the RP device and prepares to send data.

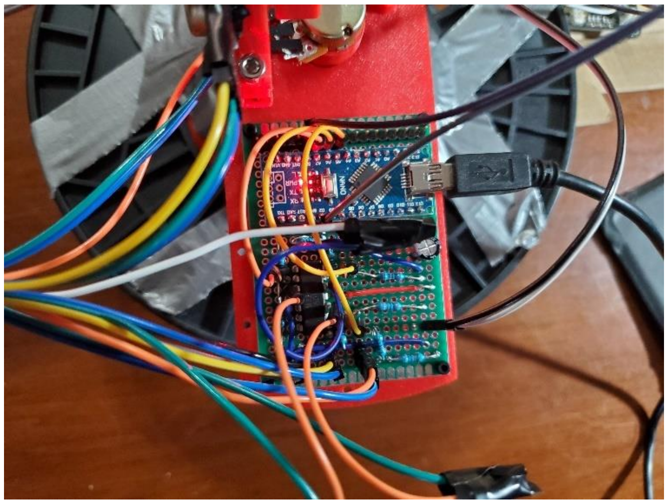


Figure 18: BERT controller

The pot sensors are powered when the PSU is activated. They will send a voltage between 0 and 2.5V depending on their angle. This voltage is wired directly to the BERT controller RC and op amp circuits. These circuits filter and stabilize the voltage and amplify it by a gain of 2. This modified signal is sent to the analog pins on the Nano. The Nano will read this data as an integer value between 0 and 1024. It will store this value into integer variables within its memory and average the reading over a defined period to create a consistent integer value for each potentiometer sensor (a, b, and c).

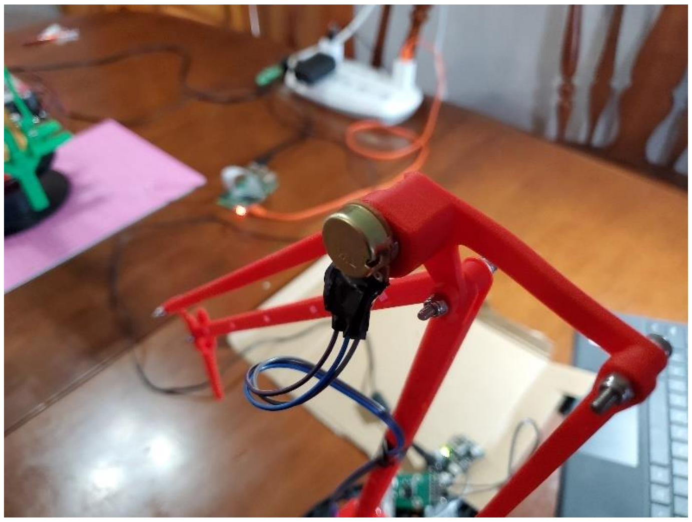


Figure 19: BERT potentiometer (position sensor)

The system will remain at this state until the RP button is pressed. When this event occurs, the Nano will send the created integer values (a, b, and c) to the RP device at a constant baud rate. When the button is released, it will cease sending the data.

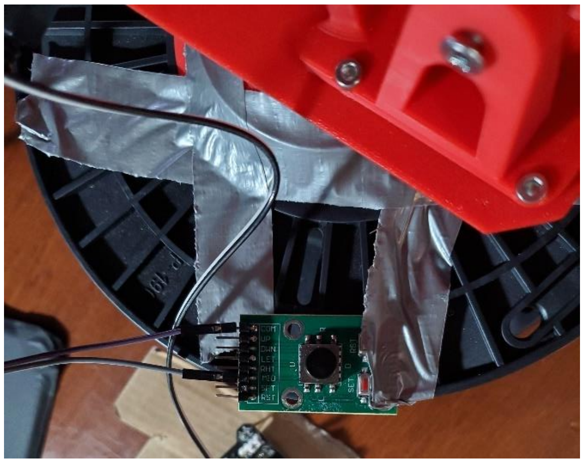


Figure 20: BERT RP interface (Button)

The RP device must then be connected to the server (raspberry pi) and the server powered on. Once a server client relationship is established, the RP device will relay the integer values (a, b, and c) from BERT’s controller to the server.

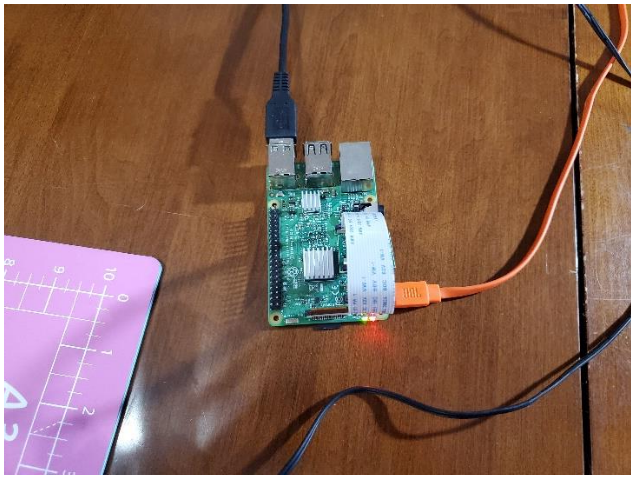


Figure 21: Server (Raspberry Pi)

The server reads the data from the RP device. The server is connected to HUE's controller via a USB port. Once the server is connected to BERT and data starts transmitting from the Nano, the server will attempt to send this data through the USB port to HUE controller.

At this point, HUE PSU must be connected to a wall outlet. The PSU will begin supply 12V to the motor driver board and 5V to the Arduino Nano in HUE's controller.

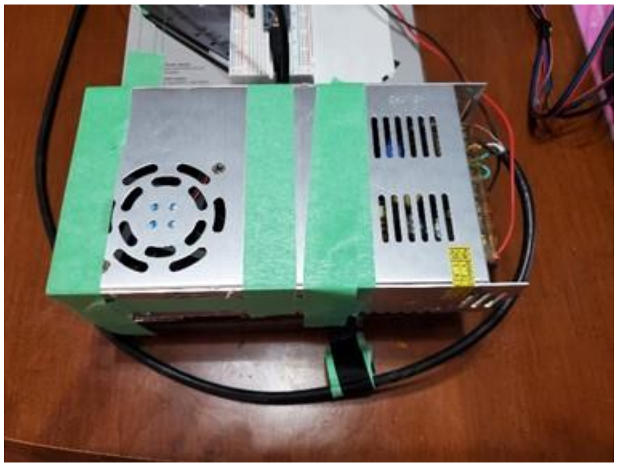


Figure 22: HUE power supply

When HUE's Nano is powered, it will boot and run set up code. It will then wait to receive serial data from the server connection. When the server relays the positional data created from BERT's sensors (a, b, and c), the Nano reads this data into variables and starts the mathematical transformation process. The variables a, b and c are fed into a mathematical function which converts the integer value into an absolute motor step orientation (S1, S2, and S3). If the receive values are not valid (HUE can't move there), the data will be discarded, and the Nano will wait for more serial data.

If the converted values S1, S2, and S3 are valid, the Arduino sets them as a target location for the stepper code objects. Next the program sends step signals (HI and LOW values to specific digital pins) and these signals transfer to the motor driver board until the target location is reached, or more serial data is received. The motor driver board receive and directs these signals to the 3 A4988 drivers which deliver the appropriate voltage to the Nema 17 stepper motors.

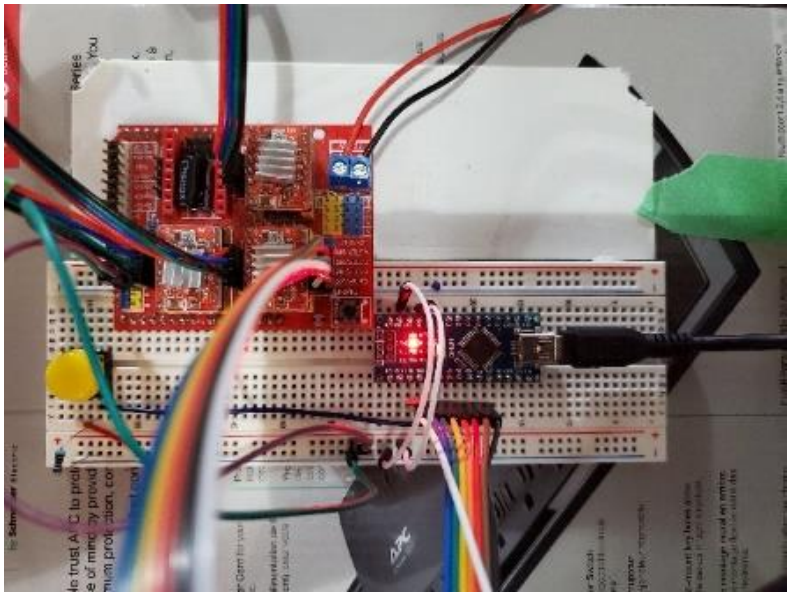


Figure 23: HUE controller (no op amps)

The Nema motors respond and drive the gearing systems on HUE's frame to move the arm.

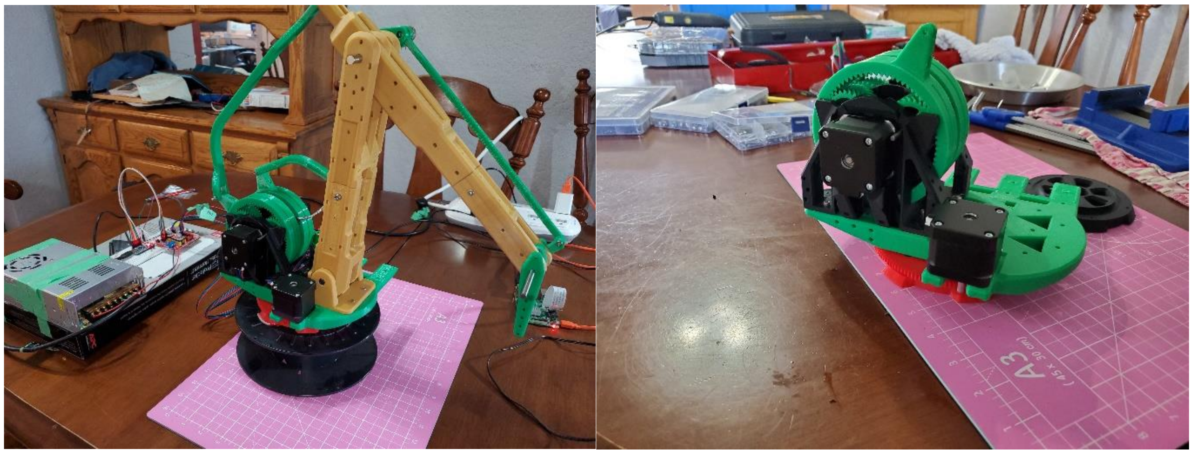


Figure 24: HUE Frame and Nema 17 Stepper Motors

At any point during when HUE and the motors are powered, the LP may press the 'disable' button. This button bypasses HUE's Arduino nano and all software components of HUE and activates the 'disable' pin on the motor driver boards. This pin prevents any electrical power to be provided to the pins of the stepper motors and leaves them inert.

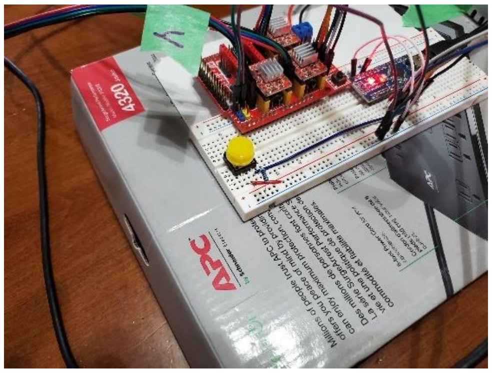


Figure 25: HUE LP interface (Button)

At this point, the RP has motion control over HUE. The RP can then move BERT to the desired location and press the RP button. HUE will attempt to move to the current location of BERT. Thus, while the activation button is pressed, HUE will mirror BERT.

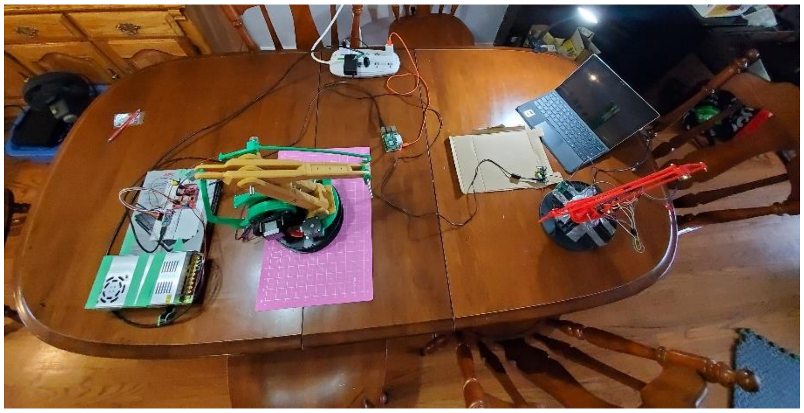


Figure 26: Complete HUEBERT system

An example of this behavior is demonstrated in the figure below. Note, when the activation button is not pressed, HUE will move to and maintain the last set target location.

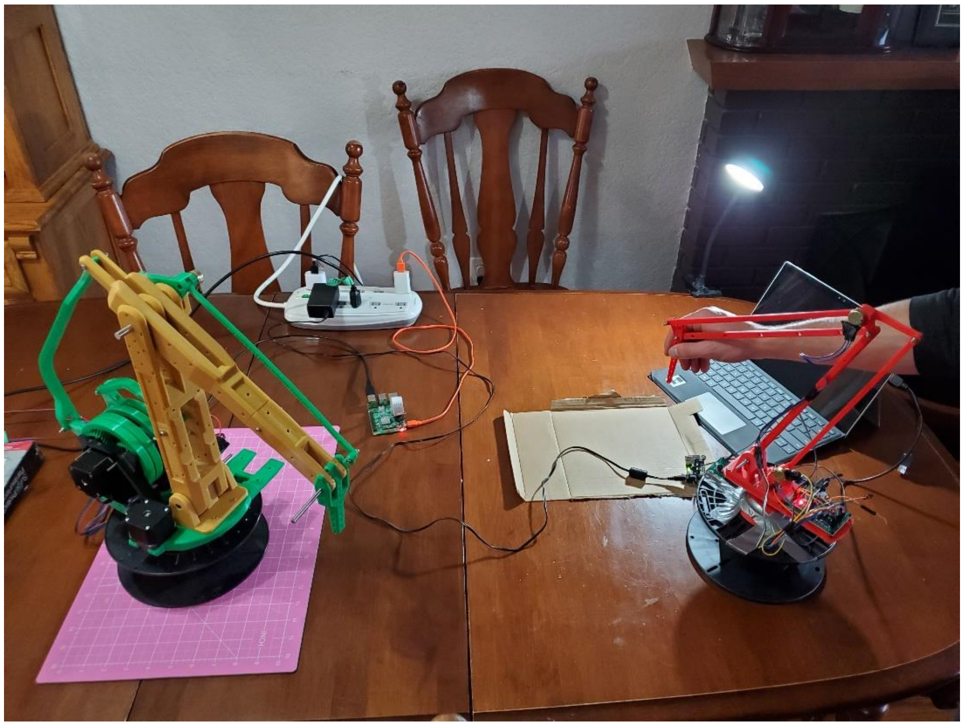


Figure 27: BERT is moved by RP; HUE mimics BERTS motion

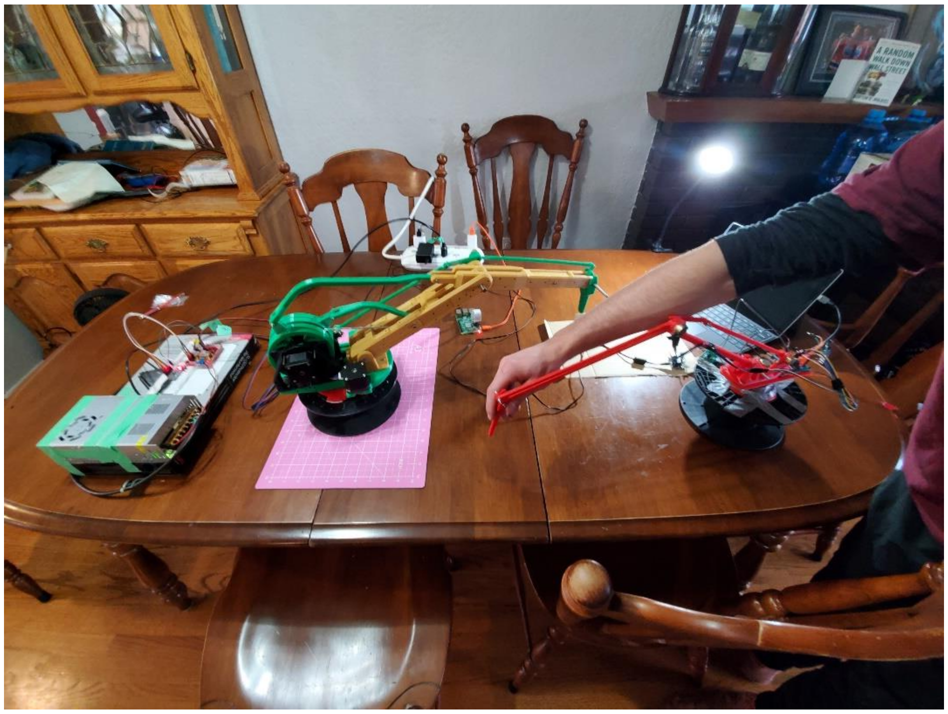


Figure 28: BERT is used to extend HUE and reach forward

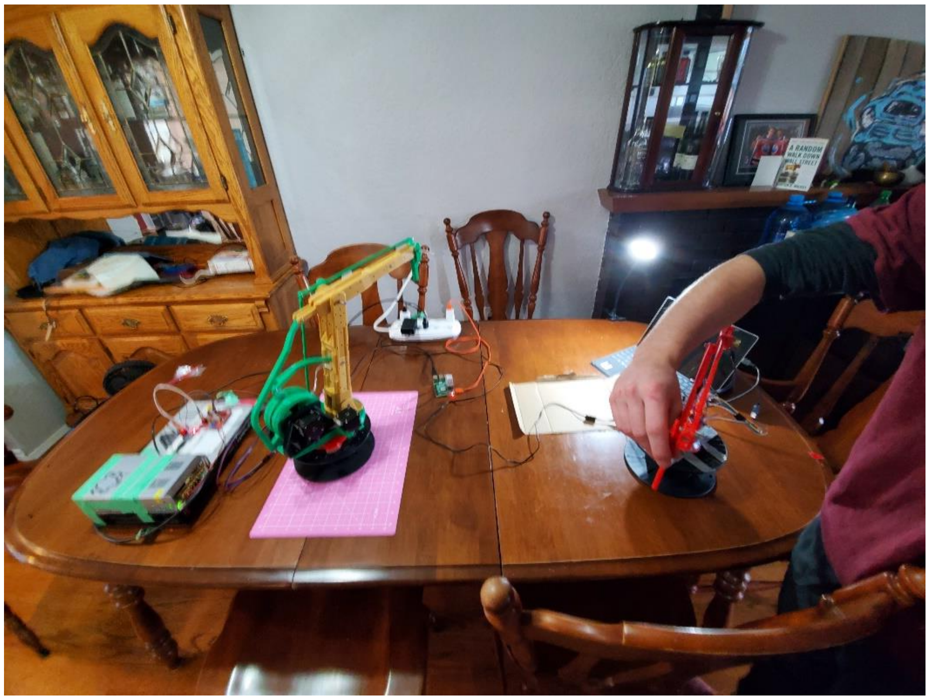


Figure 29: BERT controls HUE to lift and rotate CCW