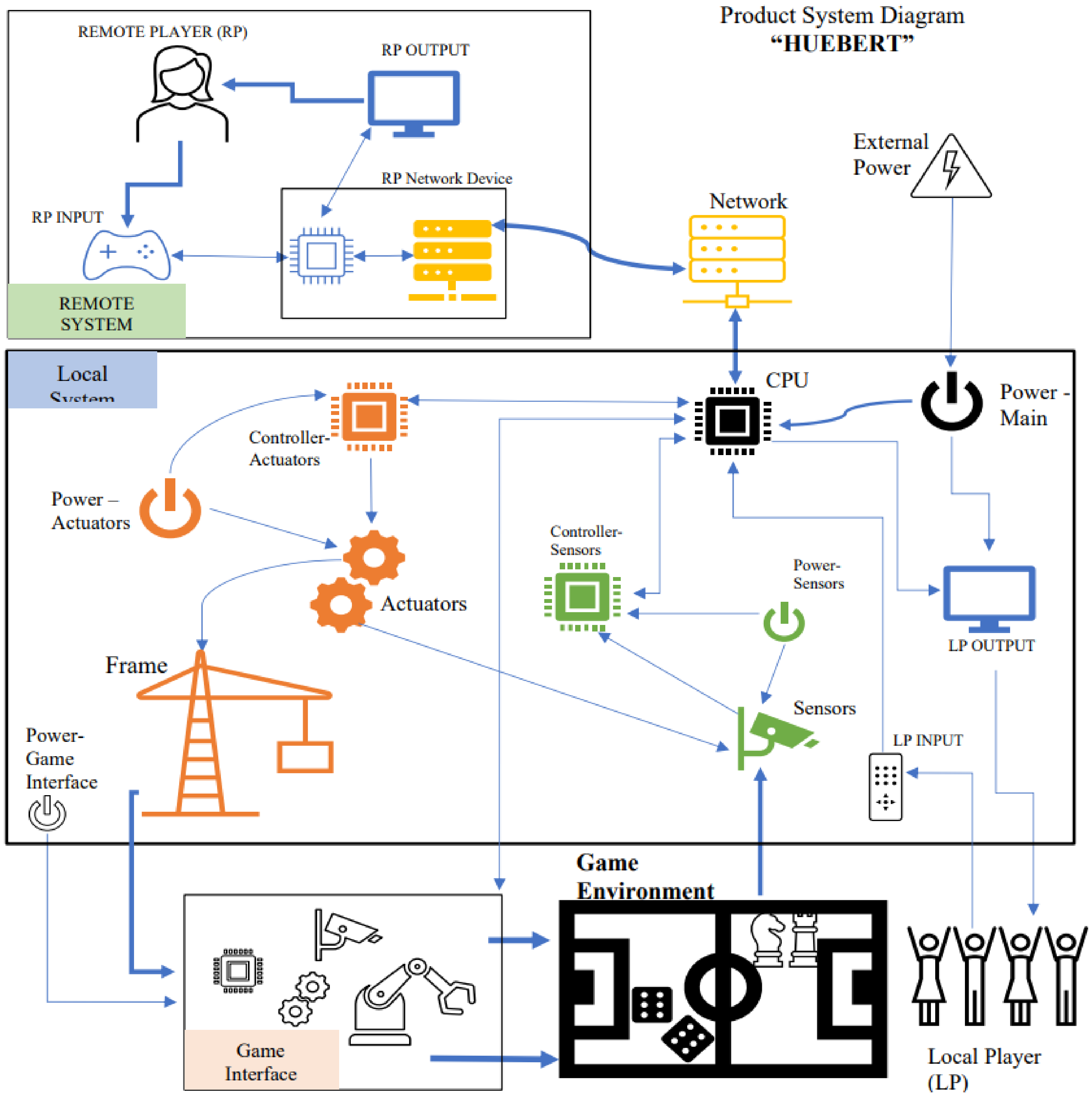
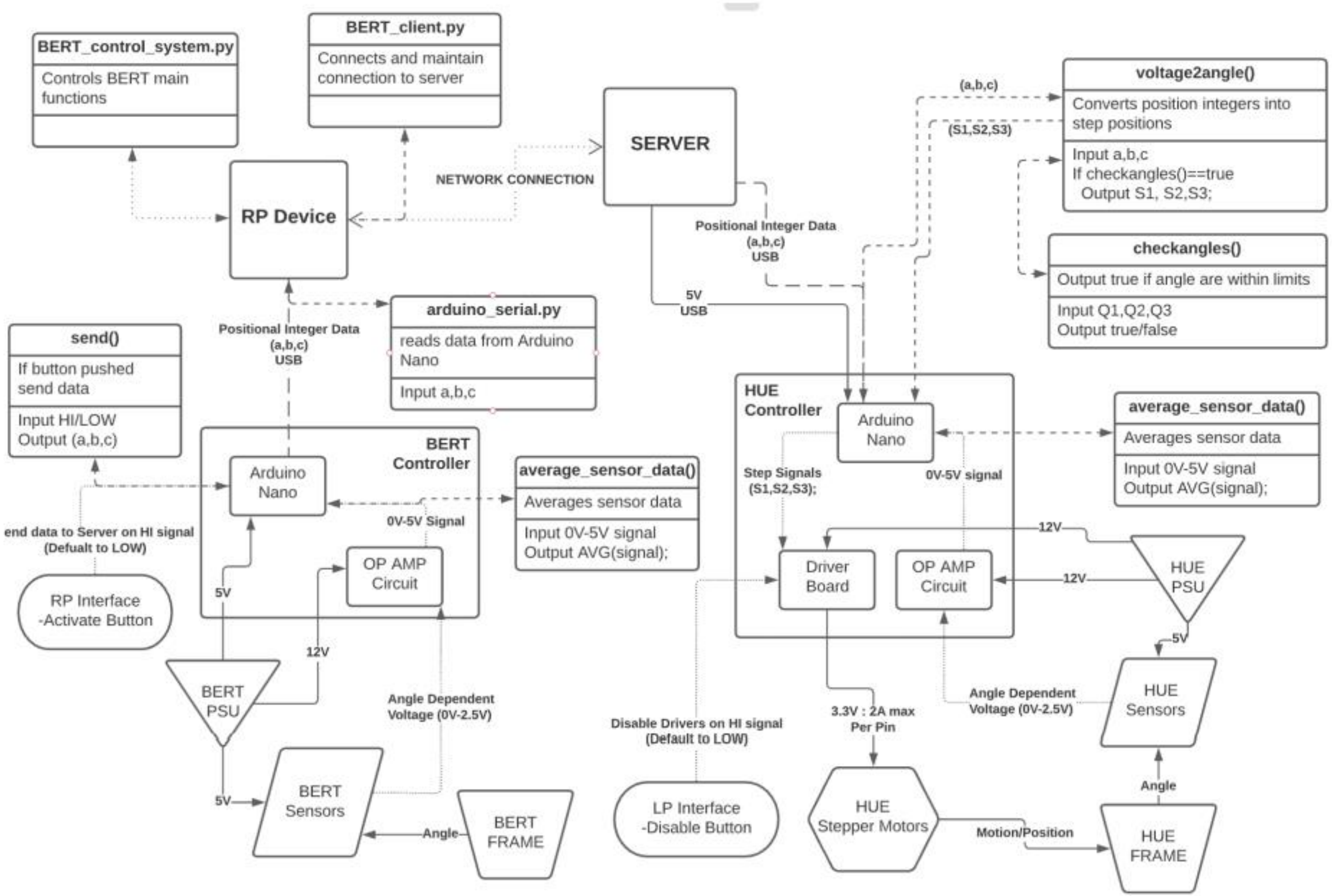
HUEBERT

A REMOTE OPERATED ROBOT ARM

System Diagram



Design Diagram



BERT_control_system.py
Controls BERT main functions

BERT_client.py
Connects and maintain connection to server

SERVER

NETWORK CONNECTION

RP Device

Positional Integer Data
(a,b,c)
USB

arduino_serial.py
reads data from Arduino Nano
Input a,b,c

send()
If button pushed send data
Input HI/LOW
Output (a,b,c)

end data to Server on HI signal
(Defualt to LOW)

RP Interface
-Activate Button

BERT Controller
Arduino Nano
OP AMP Circuit
0V-5V Signal

average_sensor_data()
Averages sensor data
Input 0V-5V signal
Output AVG(signal);

5V
12V
BERT PSU
5V
BERT Sensors
Angle Dependent Voltage (0V-2.5V)
Angle
BERT FRAME

Positional Integer Data
(a,b,c)
USB

5V
USB

(a,b,c)
(S1,S2,S3)

voltage2angle()
Converts position integers into step positions
Input a,b,c
If checkangles()==true
Output S1, S2,S3;

checkangles()
Output true if angle are within limits
Input Q1,Q2,Q3
Output true/false

HUE Controller
Arduino Nano
Step Signals
(S1,S2,S3);
0V-5V signal
Driver Board
OP AMP Circuit

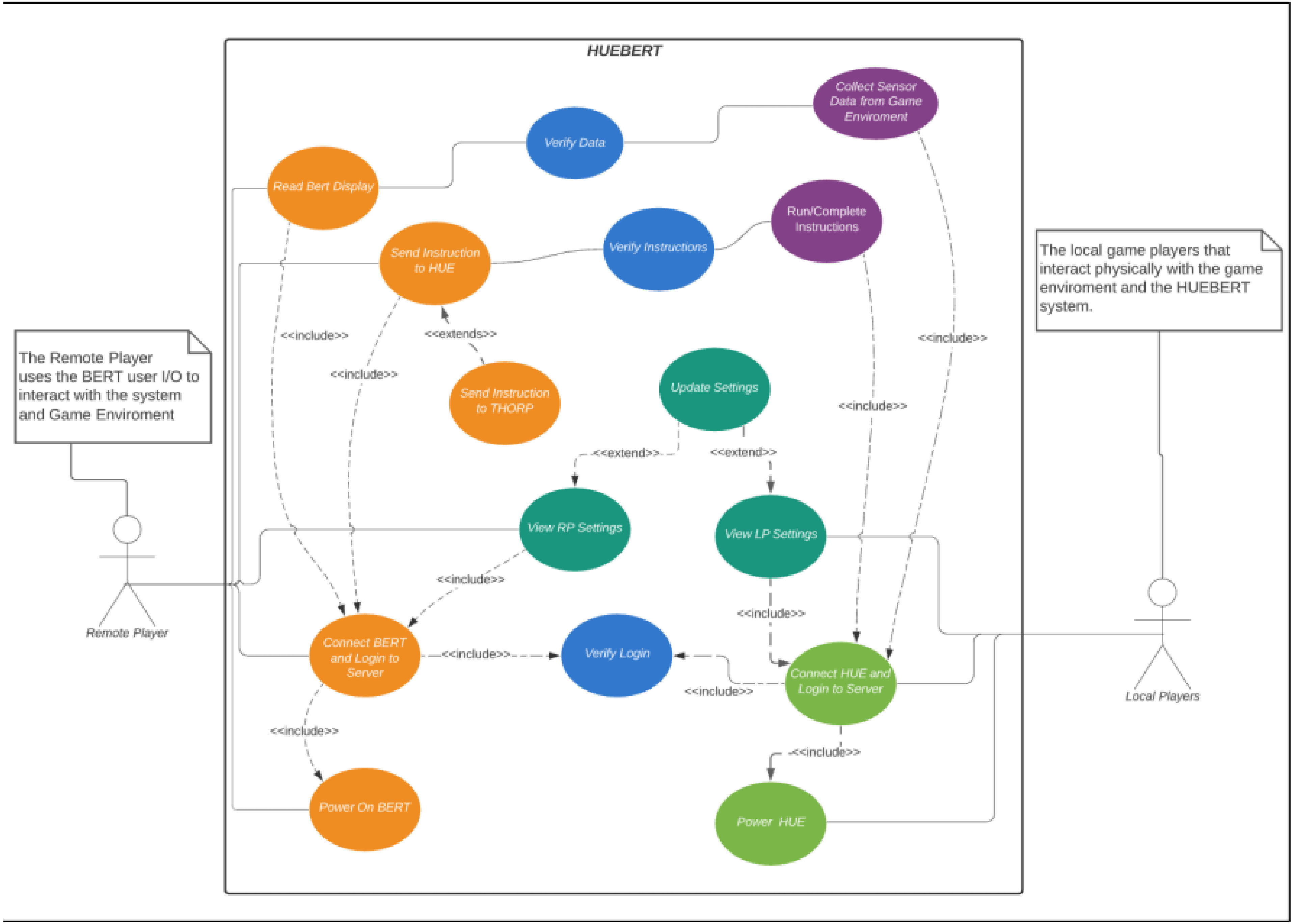
average_sensor_data()
Averages sensor data
Input 0V-5V signal
Output AVG(signal);

12V
12V
HUE PSU
5V
HUE Sensors
Angle Dependent Voltage (0V-2.5V)
Angle
HUE FRAME
Motion/Position
HUE Stepper Motors
3.3V : 2A max
Per Pin

Disable Drivers on HI signal
(Default to LOW)

LP Interface
-Disable Button

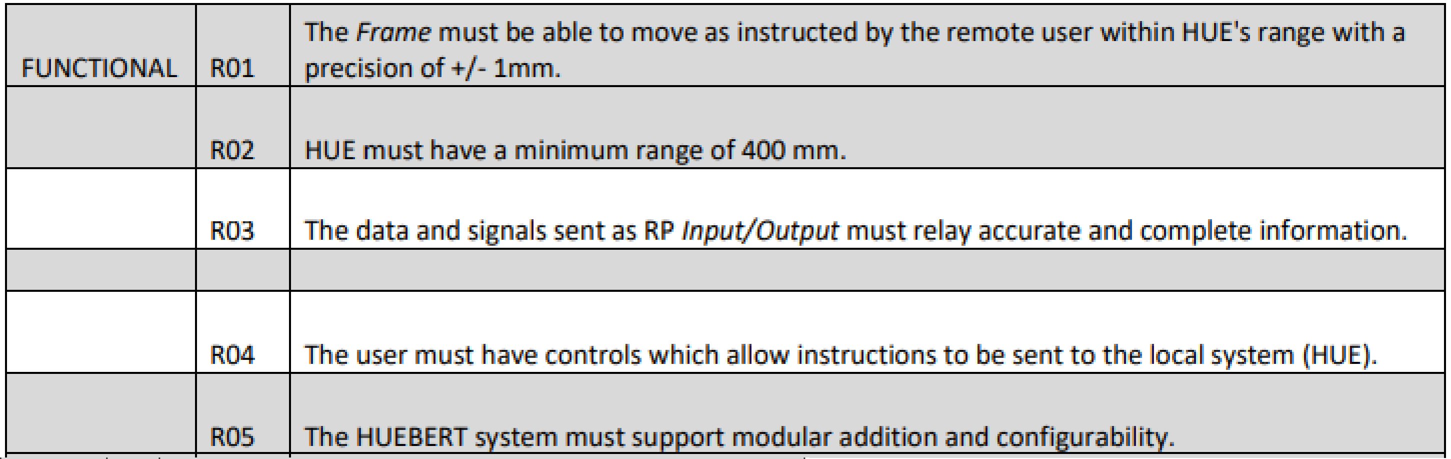
Use Case Diagram



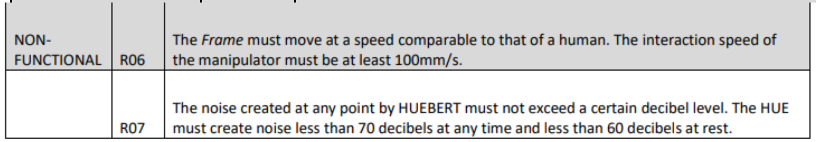
The Remote Player uses the BERT user I/O to interact with the system and Game Enviroment

The local game players that interact physically with the game enviroment and the HUEBERT system.

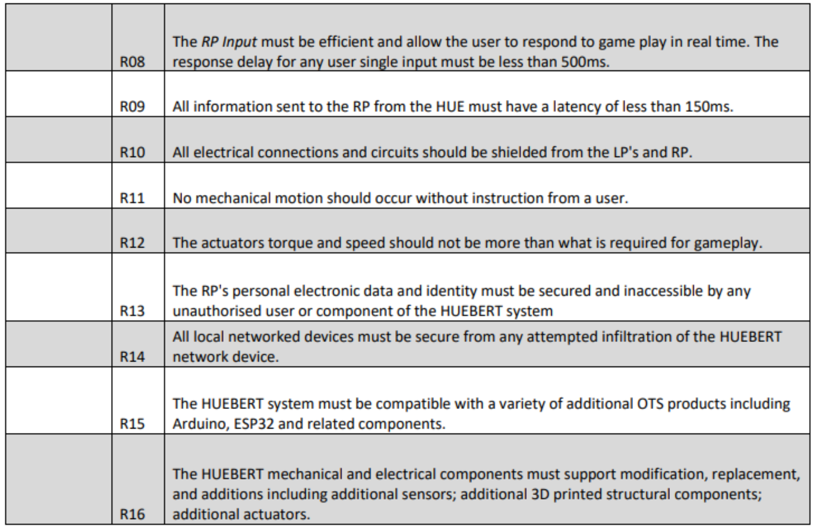
Requirements



FUNCTIONAL	R01	The *Frame* must be able to move as instructed by the remote user within HUE's range with a precision of +/- 1mm.
	R02	HUE must have a minimum range of 400 mm.
	R03	The data and signals sent as RP *Input/Output* must relay accurate and complete information.
	R04	The user must have controls which allow instructions to be sent to the local system (HUE).
	R05	The HUEBERT system must support modular addition and configurability.

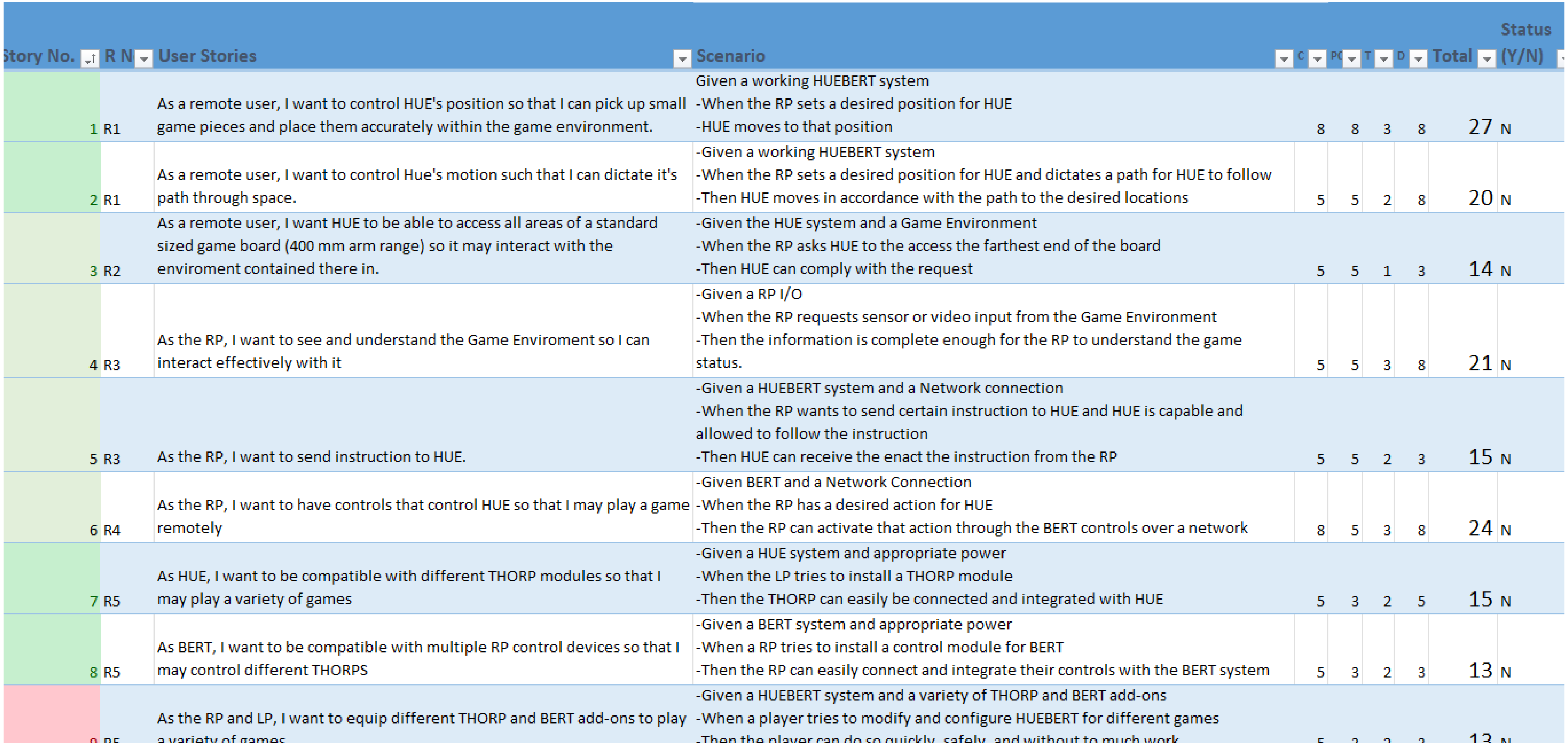


NON-FUNCTIONAL	R06	The *Frame* must move at a speed comparable to that of a human. The interaction speed of the manipulator must be at least 100mm/s.
	R07	The noise created at any point by HUEBERT must not exceed a certain decibel level. The HUE must create noise less than 70 decibels at any time and less than 60 decibels at rest.



	R08	The *RP Input* must be efficient and allow the user to respond to game play in real time. The response delay for any user single input must be less than 500ms.
	R09	All information sent to the RP from the HUE must have a latency of less than 150ms.
	R10	All electrical connections and circuits should be shielded from the LP's and RP.
	R11	No mechanical motion should occur without instruction from a user.
	R12	The actuators torque and speed should not be more than what is required for gameplay.
	R13	The RP's personal electronic data and identity must be secured and inaccessible by any unauthorised user or component of the HUEBERT system
	R14	All local networked devices must be secure from any attempted infiltration of the HUEBERT network device.
	R15	The HUEBERT system must be compatible with a variety of additional OTS products including Arduino, ESP32 and related components.
	R16	The HUEBERT mechanical and electrical components must support modification, replacement, and additions including additional sensors; additional 3D printed structural components; additional actuators.

User Stories



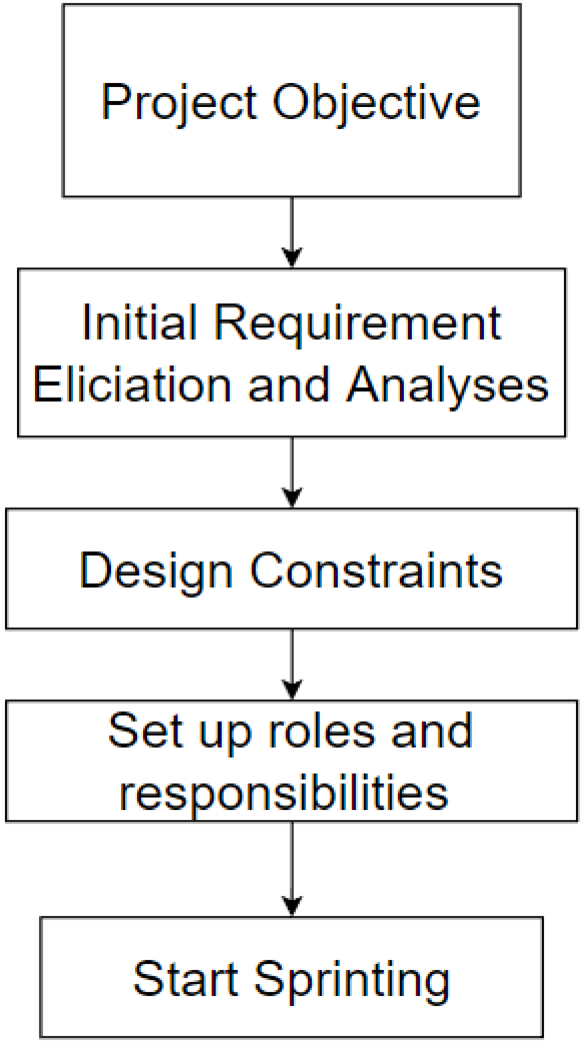
Story No.	R N	User Stories	Scenario	C	PC	T	D	Total	Status (Y/N)
1	R1	As a remote user, I want to control HUE's position so that I can pick up small game pieces and place them accurately within the game environment.	Given a working HUEBERT system -When the RP sets a desired position for HUE -HUE moves to that position	8	8	3	8	27	N
2	R1	As a remote user, I want to control Hue's motion such that I can dictate it's path through space.	-Given a working HUEBERT system -When the RP sets a desired position for HUE and dictates a path for HUE to follow -Then HUE moves in accordance with the path to the desired locations	5	5	2	8	20	N
3	R2	As a remote user, I want HUE to be able to access all areas of a standard sized game board (400 mm arm range) so it may interact with the enviroment contained there in.	-Given the HUE system and a Game Environment -When the RP asks HUE to the access the farthest end of the board -Then HUE can comply with the request	5	5	1	3	14	N
4	R3	As the RP, I want to see and understand the Game Enviroment so I can interact effectively with it	-Given a RP I/O -When the RP requests sensor or video input from the Game Environment -Then the information is complete enough for the RP to understand the game status.	5	5	3	8	21	N
5	R3	As the RP, I want to send instruction to HUE.	-Given a HUEBERT system and a Network connection -When the RP wants to send certain instruction to HUE and HUE is capable and allowed to follow the instruction -Then HUE can receive the enact the instruction from the RP	5	5	2	3	15	N
6	R4	As the RP, I want to have controls that control HUE so that I may play a game remotely	-Given BERT and a Network Connection -When the RP has a desired action for HUE -Then the RP can activate that action through the BERT controls over a network	8	5	3	8	24	N
7	R5	As HUE, I want to be compatible with different THORP modules so that I may play a variety of games	-Given a HUE system and appropriate power -When the LP tries to install a THORP module -Then the THORP can easily be connected and integrated with HUE	5	3	2	5	15	N
8	R5	As BERT, I want to be compatible with multiple RP control devices so that I may control different THORPS	-Given a BERT system and appropriate power -When a RP tries to install a control module for BERT -Then the RP can easily connect and integrate their controls with the BERT system	5	3	2	3	13	N
9	R5	As the RP and LP, I want to equip different THORP and BERT add-ons to play a variety of games	-Given a HUEBERT system and a variety of THORP and BERT add-ons -When a player tries to modify and configure HUEBERT for different games -Then the player can do so quickly, safely, and without to much work	5	3	2	3	13	N

HUEBERT DESIGN/BUILD PROCESS

What actually happened.

Could have been better/ more by the book

PROJECT SETUP



Project Objective

What do you want to make?

Initial Requirement Eliciation and Analyses

What do I want to happen?

Simulation and Brainstorm

Design Constraints

What do I have to work with?

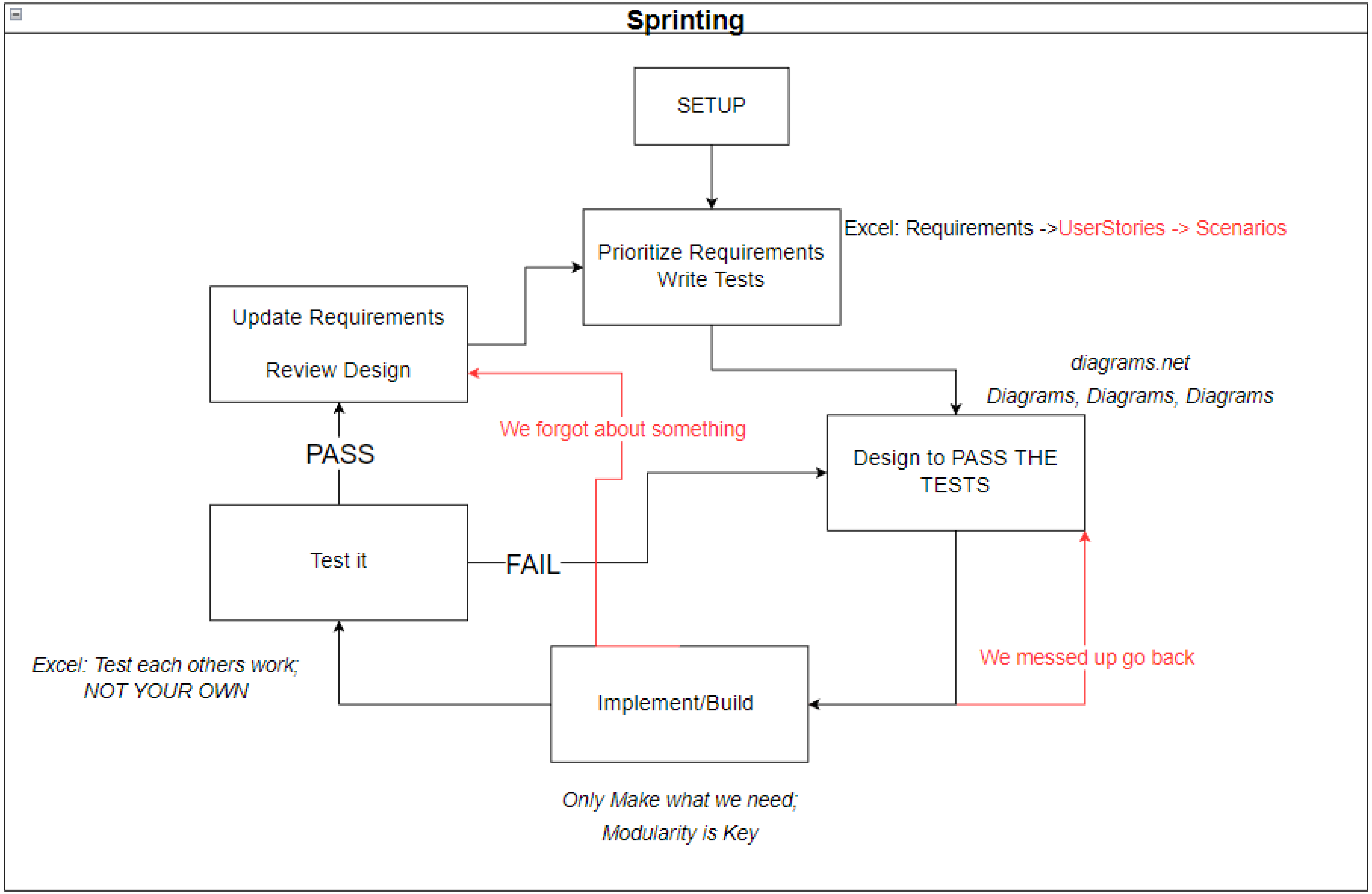
What can I order/print/make?

Set up roles and responsibilities

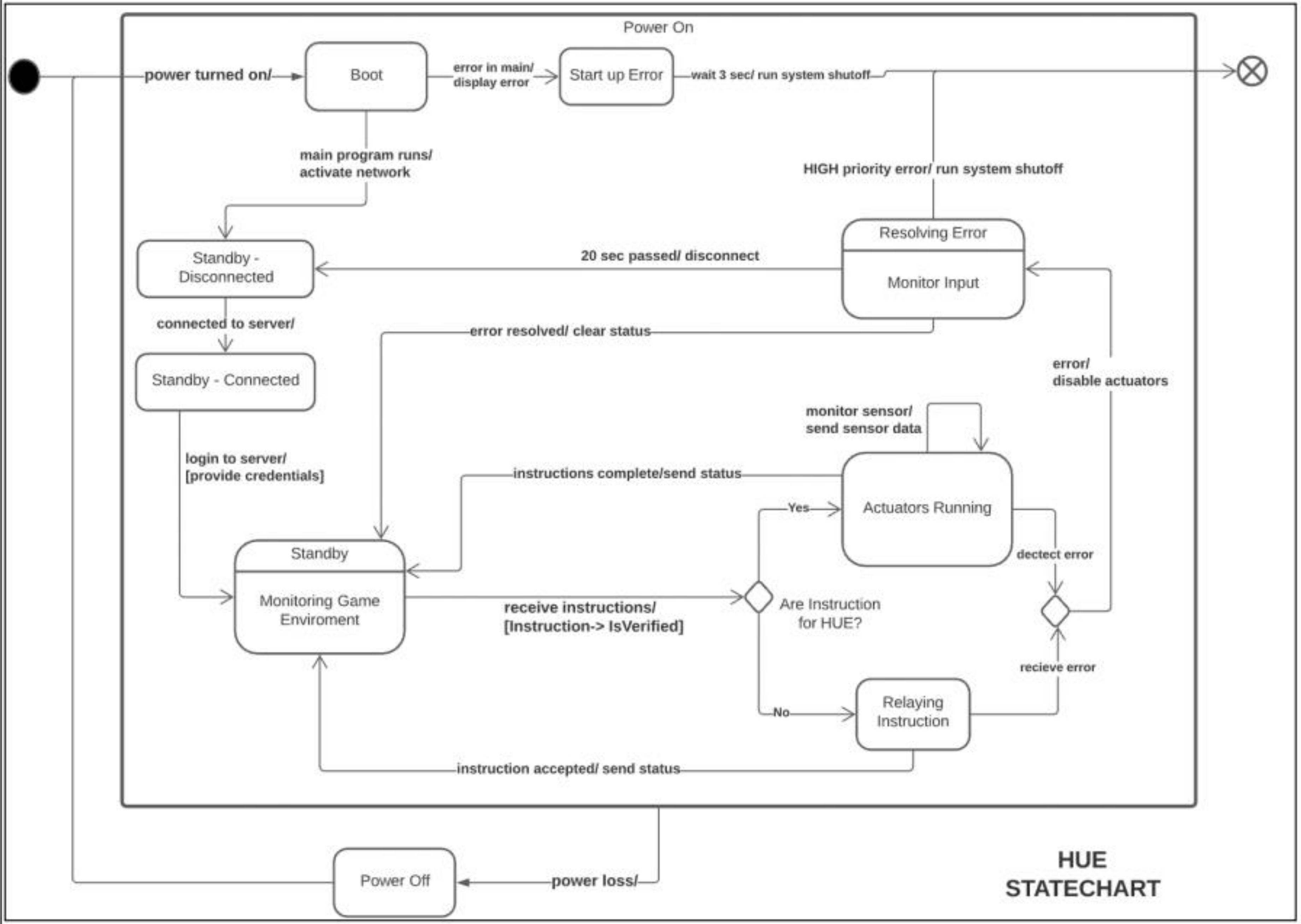
Only two of us, but good practice.

Start Sprinting

HUEBERT DEVELOPMENT PROCESS

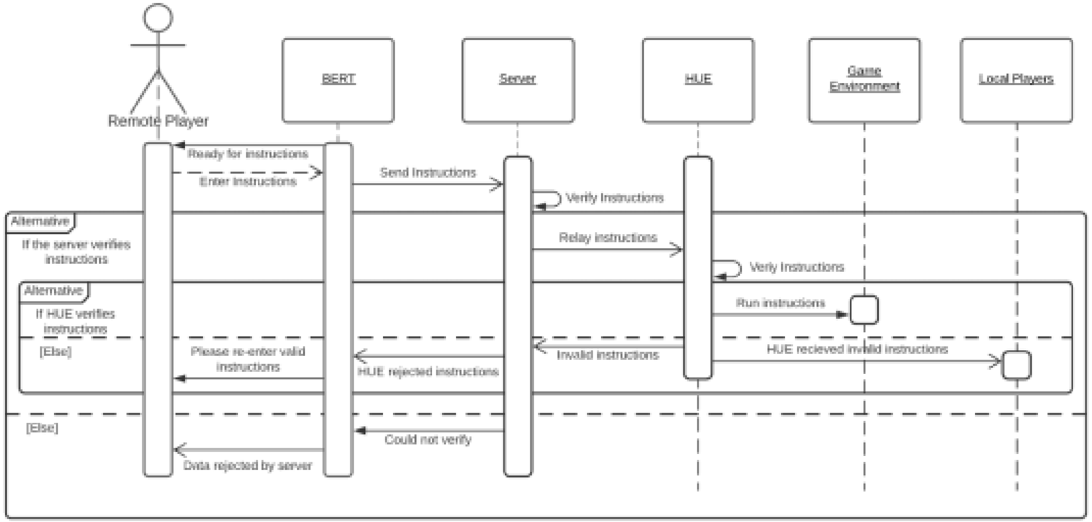


Diagrams

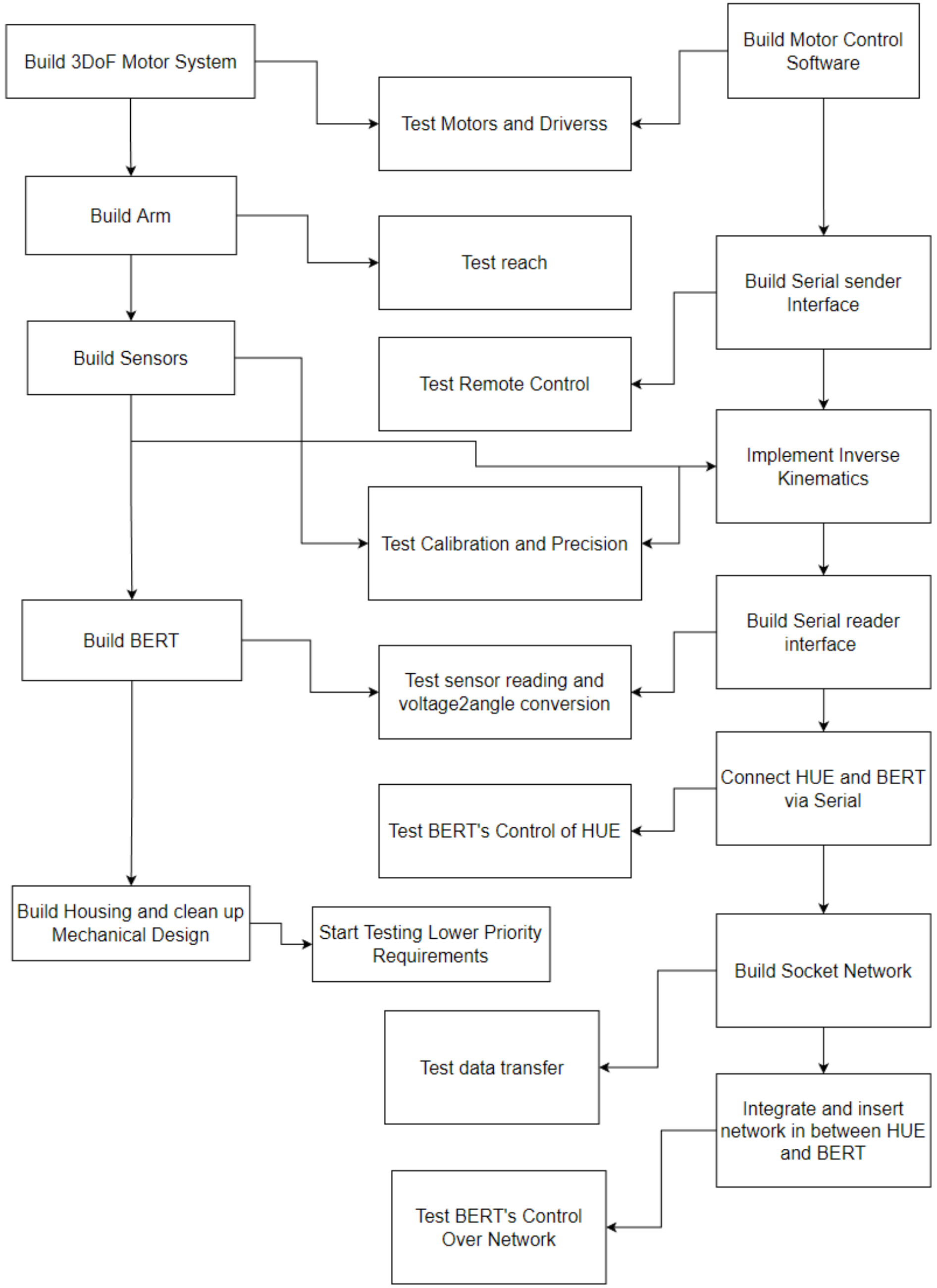

Power On
power turned on/
Boot
error in main/
display error
Start up Error
wait 3 sec/ run system shutoff
main program runs/
activate network
HIGH priority error/ run system shutoff
Resolving Error
Monitor Input
Standby -
Disconnected
20 sec passed/ disconnect
connected to server/
error resolved/ clear status
Standby - Connected
error/
disable actuators
monitor sensor/
send sensor data
login to server/
[provide credentials]
instructions complete/send status
Actuators Running
Yes
dectect error
Standby
Monitoring Game
Enviroment
receive instructions/
[Instruction-> IsVerified]
Are Instruction
for HUE?
recieve error
No
Relaying
Instruction
instruction accepted/ send status
Power Off
power loss/
HUE
STATECHART

2.1.4 Send Instruction to HUE

ID	UC-05
Description	The RP sends instruction for HUE through BERT. These instructions allow HUE to interact with the Game Environment.
Actors	Remote Player
Secondary Use Cases	UC-04: RP Login
Preconditions	HUEBERT is powered on and connected, the RP and LP have logged in to the server.
Main Flow	1. The RP enters their instruction for HUE through BERT 2. BERT compiles and sends these instructions to the server 3. If the server can verify BERT's data a. The server relays the data for BERT 4. If the controls information is verified by HUE. a. HUE runs the instructions b. HUE moves and the Game Environment changes in Accordance with these instructions.
Postconditions	HUE completes the instructions sent to it from BERT.
Alternative Flow(s)	3b. If the server cannot verify the data: a) The data is rejected, and no information is sent through to HUE b) The error message is displayed for the RP 4b. If the control information cannot be verified by HUE: a) HUEBERT notifies the RP and LP the controls were invalid. b) HUEBERT prompts the RP to re-enter valid controls (Continue from 1).



Summary Timeline



Build 3DoF Motor System

Build Arm

Build Sensors

Build BERT

Build Housing and clean up Mechanical Design

Test Motors and Driverss

Test reach

Test Remote Control

Test Calibration and Precision

Test sensor reading and voltage2angle conversion

Test BERT's Control of HUE

Start Testing Lower Priority Requirements

Test data transfer

Test BERT's Control Over Network

Build Motor Control Software

Build Serial sender Interface

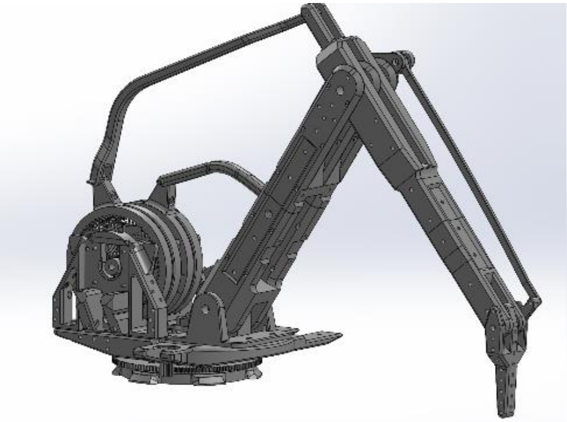
Implement Inverse Kinematics

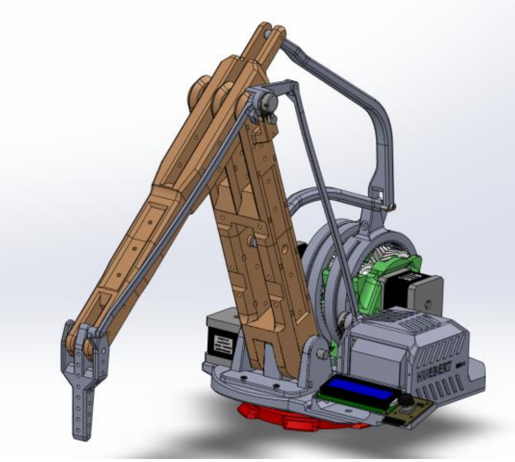
Build Serial reader interface

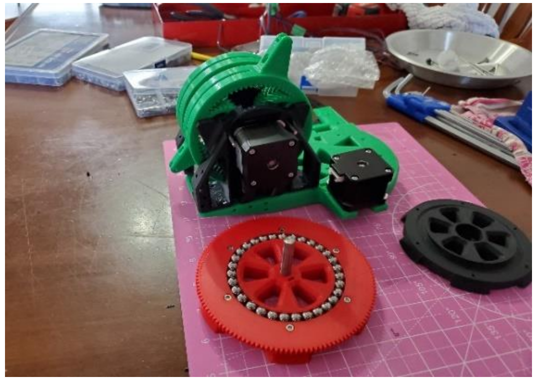
Connect HUE and BERT via Serial

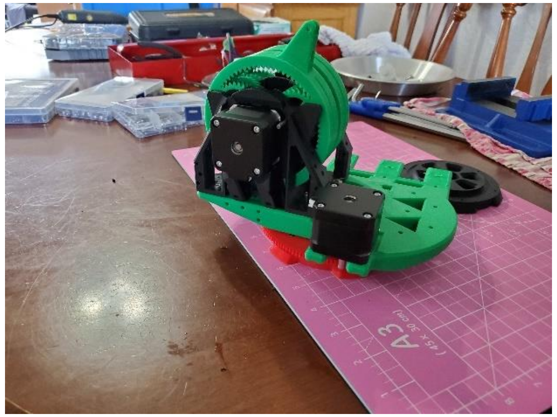
Build Socket Network

Integrate and insert network in between HUE and BERT

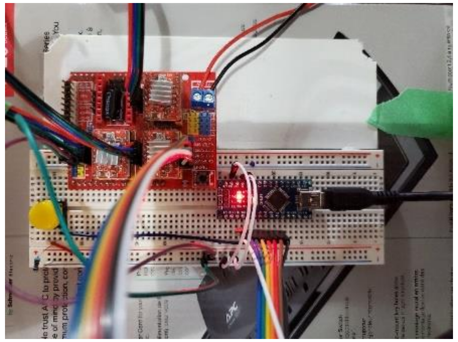


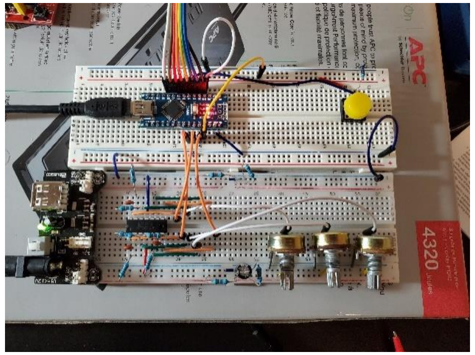

HUEBERT

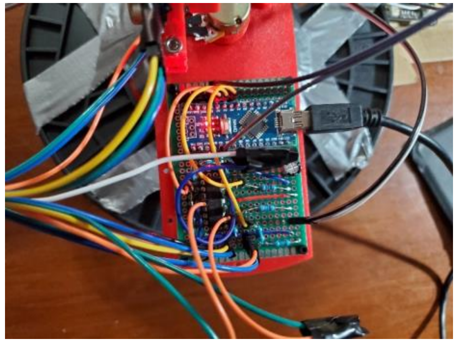


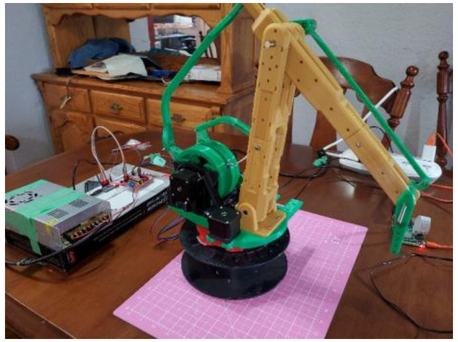

A3

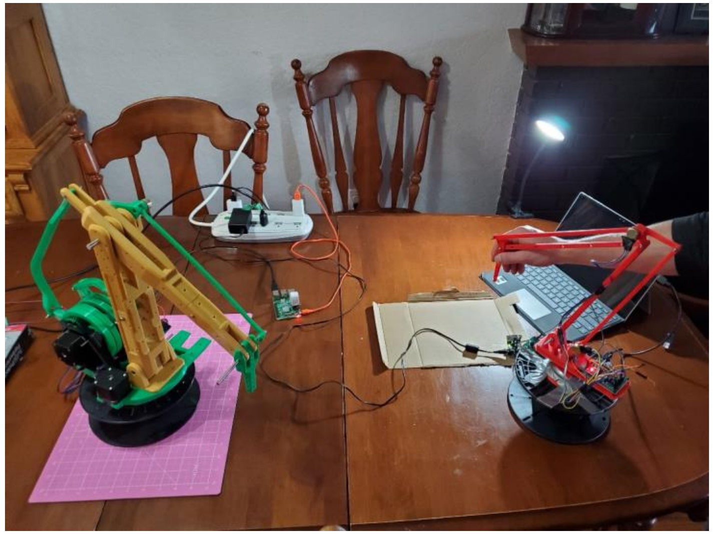
A3




APC
4320







Recommendations

What we did right/wrong

What worked well for us

- **Test Driven Development:** Always had a ‘working’ product; Less ‘extra’ work
- **Constant Hardware-Software Integration:** When hardware ready, software was tested on it
- **Diagrams:** Most valuable design step (for me at least): Made building like working from an instruction booklet
- **Agile:** We found mistakes or requirement problems all the time; Quick meeting-> change excel sheet -> update diagram -> Good to go
- **Maximize Documentation effectiveness:** We used 2 documents: *Requirements/Testing Spread Sheet, Diagram Folder*; Easy to keep up to date without spending tons of time on it. Everything else made when needed.
- **MODULARIZE:** This made things so easy to change
- **Do it EARLY:** Always busy at the end of the semester

What we should have done better

- **Make requirements/tasks smaller**
  - Always more to each step than we realized
  - Only two of us
  - Make more specific requirements
  - Take only a couple tasks each sprint
  - Hit the ‘Review’ portion of the design more often but make it shorter.
- **User Stories**
  - We didn’t use these enough; ran into issues (ie. Button hold)
- **Research:**
  - Ran into limitations with certain components (ie. Nano processing speed for inverse kinematics)
- **Prioritize Requirements/Backlog Better:**
  - Some requirements are dependent on other requirements, fit this into the prioritization process. (We had to break script because we failed to account for this).
- **Don’t get attached to a design:** I tried to force a bad solution which ended up wasting time. (ie. Use of raspberry pi; Should have bought an ethernet shield for Arduino instead.)

Effective Tools We used

- **Diagrams.net**: Easy, integrates with github and google drive
- **Shared Excel/Sheets:** for requirements
- **SolidWorks**
- **KiCad**
- **Arduino for simple embedded stuff**. Fast because of IDE, peripherals, and documentation. Big learning curve for ESP and STM chips (Tried at start but took too long).