\*\*The P3N\*\* About the P3N =============================================================================== The P3N aims to allow users to create drawings in 3D space through the use of intuitive hand motions that guide a virtual drawing utensil. This goes beyond the scope of drawing on a touchpad or using a CAD modeling software. Here the user turns their 3D hand motion input into a 3D drawing output that they can observe in a web app that hosts the UI for the project. To draw with the P3N, the user can simply press the drawing button (located on the right of the set up) and the P3N will begin to move in the forward direction on the screen. The P3N has a set velocity and the direction can be controlled through primarily tilting movement that is sensed using an IMU. Tilting to the right changes the direction of movement to the right, and the same logic to move to the left. More gradual tilting translates into curves on the screen whereas fast tilting will result in more perpendicular movements. The user has access to multiple speech-enabled commands that allow for brush customization and object placement among other things. The System =============================================================================== State and Block Diagrams ------------------------------------------------------------------------------- We began by connecting the IMU, ESP32, microphone, and buttons as follows: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* ^ ^ \* \* .-----|------. .------|------. \* \* | GND | | GND | \* \* | (DRAWING) | | (SPEECH) | \* \* | BUTTON 1 | | BUTTON 2 | \* \* | | .----------> | \* \* | ^ | | | | \* \* '--|---------' | '-------------' \* \* | .-------|----------------------------------------------. \* \* | | | | \* \* \* '-----|-----. '--. .-------------------------. | | \* \* | | | | .---------------. | | .--------|--------. \* \* .--|-----|----|----------|------|--------. | | | | 3V3 | \* \* | | | | | | '-----> OUT | \* \* | SVP I05 I016 I014 I015 | | | | MICROPHONE | \* \* | | | | | | \* \* <---- GND ESP32 3V3 ----\* | | | GND | \* \* | | | | '--------|--------' \* \* | I023 IO22 IO21 I018 I02 I04 | | | | \* \* | ^ ^ ^ ^ ^ ^ | | | v \* \* '---|-----|-----|-----|----|----|--------' | | \* \* | | | | | | '--|----------------------. \* \* | | | | | | | | \* \* | | | | | '------------------|-----------------. | \* \* | | | | '-----------------------|-----------. | | \* \* '-----|-----|-----|----------------------------|-------. | | | \* \* | | '----------------------------|--. | | | | \* \* | | .-' | | | | | \* \* .--------|-----|------------. .-----|----|----|---|-----|----|----. \* \* | | | LED SCK SDA A0 RESET CS | \* \* | SCL SDA | | | \* \* <------ GND VCC ---\* <----- GND VCC -----\* \* \* | IMU | | LCD | \* \* | | | | \* \* '---------------------------' '-----------------------------------' \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* .----------------. .---------------------. .-------------------------------. \* \* | STATE 1 | long button (1) press | STATE 2 | long button (1) press | STATE 3 | \* \* | Initial state ----------------------------> Username state -------------------------------> (Drawing mode selection state| drawing/tutorial selection \* \* | | | (Uses IMU to tilt | | user selects which mode they ---------------------------------. \* \* '----------------' .------------------> and type username) | .-----------------> want to use their ESP in | and short button (1) press | \* \* | '---------^-----------' | | from a selection of options)| | \* \* | | | '--------------|----------------' | \* \* | .-----------------------------------------' | | \* \* change | | change | | | \* \* username option | | drawing mode '--------------------------------------------------------' | \* \* and button (1) press | | option and multiuser selection and | \* \* | | button (1) press short button (1) press | \* \* .------------------------------------------------. .------------------------------------. .----------|-----------------------------------. \* \* | STATE 6 | | STATE 5 | | v STATE 4 | \* \* | Menu state | double tap button (1) | Draw State | long button (1) press | URL state | \* \* | (displays a multitude of options to interact |<--------------------------------| (displays size shape and color <---|----------------------------| (displays the URL of the web page that | \* \* | with the ESP32 such as changing the speed |-------------------------------->| of line being drawn) | | the user should go to to see their drawing) | \* \* | of your drawing, switching between moving | long button (1) press '----------|------------|------------' '----------------------------------------------' \* \* | your cursor and drawing a line) | button (1) | | button (2) \* \* '------------------------------------------------' press | | press \* \* | | \* \* .---------v-------. .--v-------------------. \* \* |send drawing data| | say a command that | \* \* | to server | | will be sent to the | \* \* '-----------------' | server | \* \* '----------------------' \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* High Level Description of Our System ------------------------------------------------------------------------------ This system consists of an ESP/Arduino that senses things such as speech and motion. It converts motion senses to estimated position values. It also uses an LCD to give the user some options as to what they can do. The ESP then puts all of this information into two databases that are located on the server. One of them only stores the positions and it stores x, y, z positions which are collected every so often and only when the values are like different enough by some condition. The ESP also puts the speech command (once it has been interpreted by the Google API) into the command database in its string format. ESP System ------------------------------------------------------------------------------- This part of the system is responsible for collecting acceleration data and sending it to the server. An inertial measurement unit measures the three dimensional acceleration of the device, which is then converted to a coordinate point that is then posted to the database on the server. There is also a microphone that listens to the user’s commands that is posted to the server as well. The LCD screen has instructions for the user so they know how to use the device and what website to go to in order to view their art. The screen also has instructions for changing settings on the drawing. The user uses a combination of tilting and pressing two buttons, one for voice commands and the other for selecting options and drawing, to draw with the device. Server System ------------------------------------------------------------------------------- The server consists of two databases, the POSITION\_DATABASE which keeps track of current position of the pen’s cursor and the COMMAND\_DATABASE which stores the most recent spoken command as an unparsed string. The ESP senses and puts information into the database whereas the web app gets and displays information from the databases in the form of visuals. The web app also makes use of this server in order to render visuals specific to a certain user, as well as load what the user has previously created. Web Application System -------------------------------------------------------------------------------- The web app is responsible for interpreting the data that is posted onto the server side database and visualizing it for the user. The most important aspect is keeping track of the user’s 3D cursor. If the user is in draw-mode the web app takes the new position datapoint and appends it to an existing line if there is one (otherwise it creates one). Another ten points in the current direction of movement are added. This allows for the drawn line to appear to be rendered seamlessly even though the points of movement are not being sampled as frequently (because of physical limitations and energy usage concerns). If we are in “move” mode, then the cursor is simply being moved and should not be drawing. In this process, the position of the cursor marker is changed and no points are being added to a line. The web app also parses the spoken commands as interpreted by the Google API and posted in the commands database. These commands need to follow a specific structure, and if spoken correctly, they can change aspects of the UI and pen properties. There is also a command to place cubes and spheres of specified sizes by the users to facilitate the construction of complicated compositions. The web app is written with HTML, Javascript and three.js which is the library that allows us to create a three dimensional canvas for the user to draw on and explore. The Code ================================================================================= Microcontroller Code ------------------------------------------------------------------------------- We began by using the Button class from exercises to distinguish between long and short button presses. There is also a Selector class that was taken from the Wikipedia exercise that allows the user to tilt the device to type out a certain word or phrase. It also has the added functionality of being able to scroll through a list of options instead of just the alphabet. This is done by keeping track of the index of the row that the pointer should be on rather than the index of the alphabet. To setup, the microcontroller connects to Wi-Fi, connects the IMU, initializes the pins for two buttons, and initializes the LCD screen. Once the setup is done, we use do\_ui() to instruct the user how to proceed to their drawing using the LCD screen. ~~~~~~ void do\_ui() ~~~~~~ This function utilizes a state machine to change the text on the screen based on button presses. - state 0: The user is directed to long press to start. The long press will cause them to move to state 1. - state 1: The user enters a username using the aforementioned scrolling technology. This involves tilting to switch characters and short pressing the button to select the character. Another long press causes a transition to state 2. - state 2: The screen then tells the user to choose the draw setting they desire: a tutorial, multiuser, or just freestlye drawing. The user again selects using the Selector class. If they choose the "join a friend" option, they must then enter the other user's username. After these steps, a short button press will lead to a screen explaining what link to go to on the user's computer. After another long press, the user moves to state 3. - state 3: The username, state (drawing or moving cursor), movement style (curvy or straight), and speed of the drawing are displayed and can be changed using the menu, which is controlled by the do\_menu() function. ![video](ui.mp4) ~~~~~~ void do\_menu() ~~~~~~ If the user double clicks the right button, it will take them to the menu, which is managed by this function. The function uses the Selector class to choose the following options: draw, move, straight lines, curvy lines, increase speed, decrease speed, change username, amd change drawing mode. Once the option is selected, the corresponding action is done using a state machine. The action is generally done by changing a specific variable that is included in the post request to the server or is involved in the drawing process handled by the do\_drawing() function. For example, to change from the draw state to the move state, the variable "DRAWING" is simply changed from "DRAW" to "MOVE." The user can now hold down the button on the right and tilt the device to draw in three dimensions! They can also use the button on the left to do the voice commands. These voice commands are essentially handled by the same code as the lab 6a, which used the Google API. We included more key phrases to the code to match the commands associated with the P3N. Menu: ![video](menu.mp4) Voice commands: ![video](mic.mp4) ~~~~~~ void do\_drawing() ~~~~~~ This function begins by reading the acceleration in three dimensions from the IMU. It begins by collecting the acceleration measurements from the IMU in all three dimensions and converts them to m/s^2. For each dimension, there is an array of size 10 where the measurements are added. These ten data points are then averaged together. If the user requests the straight movement style for their drawing, the average is plugged into the make\_straight(float num) function, which uses rounding to make the points form a straighter line. ~~~~~~ float make\_straight(float num) { float ret = 0.0; if (num < -.5) ret = -1.0; if (num >= -.5 && num < .5) ret = 0.0; if (num >= .5) ret = 1.0; return ret; } ~~~~~~ The resulting average is then plugged into the following equation to convert to a position point: ~~~~~~ x\_pos += SPEED\*sign(x\_avg)\*x\_avg\*x\_avg\*LOOP\_PERIOD/1000\*LOOP\_PERIOD/1000\*10; ~~~~~~ This equation is derived from integrating the acceleration twice to get position. The acceleration is squared in order to amplify acceleration and therefore the changes in position. The syntax of course changes a bit depending on the dimension that is being calculated. Once the positions are calculated, they are entered into the positions char array. A post request to the server is then performed with the following arguments: - user: username of the user - owner: the username of the person's drawing that the user wants to draw on (this is their own username unless they are doing a multiuser drawing) - positions: array of the three dimensional position that was calculated from acceleration - system\_state: "DRAW" or "MOVE" The post requests are posted using the non-blocking function, do\_nb\_http\_request, in order to minimize lag in the drawing. ![video](drawing.mp4) The following shows the multiuser feature: ![video](multiuser.mp4) The following shows the tutorial feature: ![video](tuto.mp4) Server Side Code ------------------------------------------------------------------------------- The code for the server follows a similar format to other databases created in previous labs. Here the POSITION\_DATABASE takes in the following arguments: - “user” - username of the person currently drawing - “user\_location” - the x, y, z position of the user’s cursor in a tuple that is transformed into a string. - “state” - could be either “draw” or “move” depending on whether the user wants to draw as they move the cursor or if they simply want to move the cursor without drawing a line. - “timing”- stores a timestamp of when the datapoint was posted The COMMAND\_DATABASE takes in the following arguments: - “user” - username of the person currently drawing - “command” - a string of the command as spoken into the ESP - “timing”- stores a timestamp of when the command was posted Web Application Code ----------------------------------------------------------------------------------- ~~~~~~ init() ~~~~~~ - Initializes everything for the user to be able to draw. This includes the makeRoom() function that creates the visual space for the user. If the user already had a drawing from a previous session then this function will get the drawing and render it before the user starts drawing. ~~~~~~ doDrawingApi(lastPostTime) ~~~~~~ - This function gets information from the position\_database. If a new point has been posted, this function will decide what to do with it and render the visual output. If the state of the point is “draw” this means that the user is moving and drawing, so the new point should be added to a geometry. Additionally, when drawing, this function will render 10 extra points in the direction of current movement so that the rendering looks seamless. If the state of the point is “move” then only the position of the cursor should be updated. If the state of the point is “stop” then that means the user has finished one geometry and the next time they draw they will start another. ~~~~~~ doCommandsApi(lastPostTime) ~~~~~~ - This function gets information from the commands\_database. If a new command has been posted, the function will parse it word by word with a switch-case, which is why the exact wording of the phrase is important. If an exact command has been found then this function will make the corresponding changes to the display. ~~~~~~ drawLine() ~~~~~~ - Here we manipulate three.js to draw a continuous line based on the input points we get from the database. (I don’t really know how to better explain this). ~~~~~~ drawSphere(user, size) ~~~~~~ - This function is called when a user says a draw sphere command. The sphere will be drawn at the location of the cursor with a given size. ~~~~~~ drawSquare(user, size) ~~~~~~ - This function is called when a user says a draw cube command. The cube will be drawn at the location of the cursor with a given size. Overcoming Design Challenges =============================================================================== Collecting Precise Position Coordinates ------------------------------------------------------------------------------- The main challenge the we had was trying to collect precise position coordinates by using the IMU. During the first week, we attempted to use the IMU's acceleration data and convert it into position data that matched the location of the device in space. Initially, we tried to use trapezoidal sums to get velocity data from several acceleration points then we used another trapezoidal sum to get position data from several velocity points. From doing this, however, we saw that most of the time the values we would get were too large. This was because the noise in the data that the IMU's accelerometer gets exponentially large when we do the trapezoidal sums. Then, we considered doing a low-pass filter to filter out the lower frequency values that would potentially interfere with our data but it still did not mitigate our issues. We also noticed that it was hard to ignore gravity when we were tracking such small movements. Since the axis that detected gravity would change depending on how the device was tilted, it was difficult to get rid of this force in our acceleration measurements. In the end, we realized that getting exact points in space was not a realistic approach given time, so we decided to use gravity to our advantage and change the direction of our drawing when we would tilt the IMU and keeping track of which axis was detecting gravity. When the IMU is not tilted, we would draw a straight line. We saw that this was the most pragmatic solution because we would track less movement and focus only on the direction that gravity is pointing. Approach to Energy Management ------------------------------------------------------------------------------- In order to preserve energy and power, we decided to power off the LCD after twenty seconds of user's inactivity to make sure that we would conserve power when the device is not being used. We also decided to only post to the server when the button is pressed rather than during the whole duration of the device being in the draw state. Finally, we ensured that we would only post every three data points, and we only post it if that data point is significantly different from the previous data point that was posted. Parts List ================================================================================ We did not use any parts outside of our base system because we're poor college students and are not used to spending money :).