GIMO

the face tracking glab-capable plywood made mechanical arm with 4 degrees of freedom

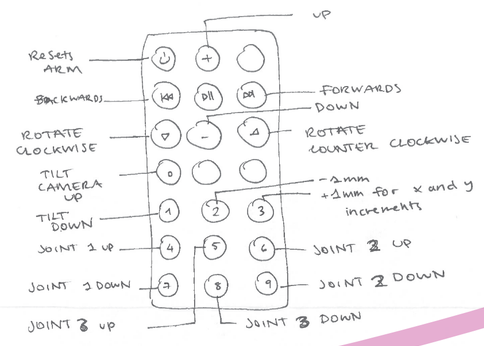
HOW-TO

STEP 1 : CONNECT 7V 1.5A power supply to the red (+) and blue (−) ground wire

STEP 2 : TURN ON HOTSPOT for the camera to connect

STEP 3 : face the camera and try to eskape its ALL SEEING nature

STEP 4 : Manually ~~control~~ the camera with the remote.



CODE

1) Inclusion of all required libraries
   - → stepper motor
   - → Wire.h for I2C communication
   - → PCA9685 (by Adafruit) for PWM control
   - → IRRemote library
   - → Servo library

2) Instantiation of all components and objects and variables
   - → IRRemote
   - → PCA driver
   - → stepper motor
   - → RGB led
   - → ESP 32 CAM

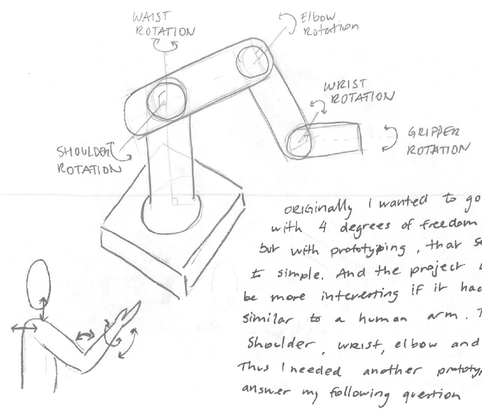
3) All helper functions
   - → stepperAngleCheck()
     - → looks at the current position of the stepper motor and determins whether or not it can physically turn or not due to the limitation of the cable
   - → set color() and color()
     - → set the RGB led to a specific color
   - → recvWithStartEndMarkers()
     - → checks if the camera has sent any data through the serial 2 pin
   - → showNewData()
     - → translates the camera data into actionable variables.
   - → InBounds()
     - → checks if a given x,y coordinate is physically allowed by the arm.
   - → printDouble()
     - → prints to the serial monitor a float / decimal number
   - → armInitialize()
     - → returns the arm to position (40, 150) and moves the stepper back to its original state.
   - → armInitialize()
     - → returns the arm to position (40, 150) and moves the stepper back to its original state.
   - → stepperAngleCheck()
     - → looks at the current position of the stepper motor and determins whether or not it can physically turn or not due to the limitation of the cable
   - → setAngle()
     - → given a servo and angle, its maps out the pre calibrated PWM value for each servo. This function also verifies that the servos do not go a position that is dangerous for the arm.
   - → moveXY()
     - → given an xy coordinate, it performs an inverse kinematic equation to determine a possible solution for the value of each angle of each servo so that the camera is level.
   - → moveX() and moveY()
     - → using moveXY(), moves the arm up/down, left/right by 5mm
   - → translateIR()
     - → given a hexadecimal value recieved from the IR remote, it acts on the order given.
4) setupFunction()
   - → initialize Serial ports
   - → setup motors and drivers
   - → armInitialize()
5) loopFunction()
   - → check if camera has sent data
   - → move camera if yes
   - → check if IR signal recieved
   - → move pwms (servo motors)

1) Initialize camera settings and pins
2) Connect to WIFI
3) FIND face
4) SEND face via serial 2 (Megaboard)

PRTOTYPING

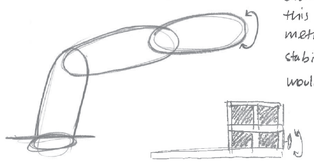
Questions I wanted Answered

How many axis of rotation do I want?



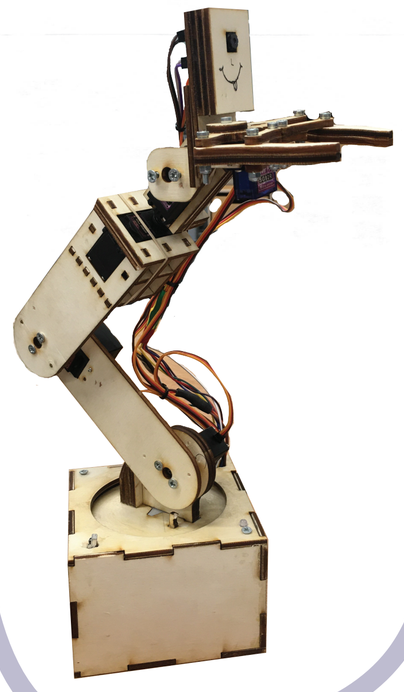
originally I wanted to go with 4 degrees of freedom but with prototyping, that seemed a simple. And the project would be more interesting if it had 5degrees similar to a human arm. The Shoulder, wrist, elbow and fingers. Thus I needed another prototype to answer my following question

How do I implement a 5th Axis of revolution? (the gripper rotation

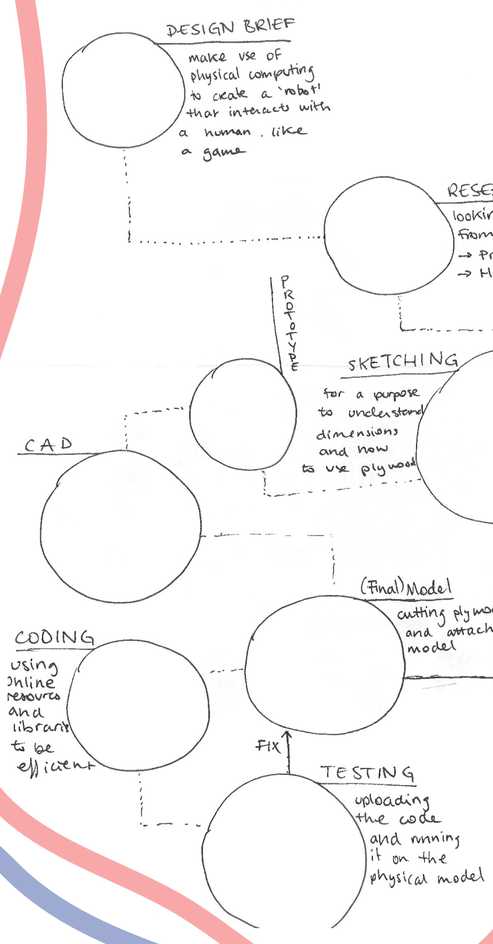


essentially, the issue with this movement was the method of holding and stabilizing the servo that would allow this rotation.

PHYSICAL MODEL



DESIGN PROCESS



PROBLEMS

~~ARM~~ ARM COUNTER WEIGHT

ARM COUNTER WEIGHT
- the center of gravity of the arm, due to the weight of the servo motors made the box fall over.
- The heaviest object I found was a broken stepper motor. everything else did not counter the weight enough

BASE PLATE
- The friction between the box and the base plate was too high so the arm could not turn.
- Adding thin layers of shellac primer helped reduce this friction but it was still an issue.

ESP 32 library
- the tutorials I found online to control the ESP 32 camera did not specify what library to download so I thought that downloading the most recent one would allow for all functionality. Yet the newest library does not contain the facial detection so crucial to my project.
- I found this out by posting to a github forum for the first time as the most recent comment was 3 days old.

~~STEPPER~~ STEPPER MOTOR ROTATIONS
- The issue with the stepper motor is the lack of control regarding absolute position. It knows how to turn but not where it is at. It cannot prevent full rotations (cables)
- To solve this, a counter was implemented and before the stepper was moved, a check verified that the stepper would remain in the bounds.

IR REMOTE
- With this remote, a long press is understood by the hex value 0xFFFFFFFF. In the original code, that corresponds to the value "repeat", but not to repeated value.
- To solve this a simple check checks for the value repeat and instead outputs the previous value pressed, stored in a variable.

BOX SIZE
- All of the components needed to fit in the box for aesthetic reasons, and a practical one of transporting and storage.
- A lot of pain went into wiring and rewiring all of the components making sure that no wires are bulkied. It was similar to a jigsaw puzzle.

ESP 32 WIFI

~~ESP 32 WIFI~~
- The device requires a ~~wifi~~ with low security on a 3.3~~/6~~GHz

ESP 32 WIFI
- The device requires a WIFI with low security on a 2.4 GHz ~~not~~ network. The imperial WIFI does not provide this as they are WPA2 protected, with a username and password, not a traditional SSID.
- I solved this by using my iPhone's hotspot (6th gen) as the hotspot generated is not 5GHz

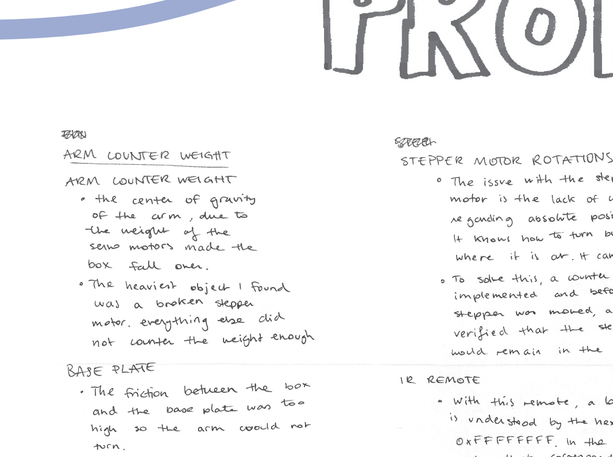
SERVO MOTORS JITTERS
- servo motors after being told what position to go to seem to jitter around, using the PCA 9685 drive fixed this issue
- another issue was when moving the servo motors using the inverse kinematics, the servos and the entire arm would jump to the next location, not at all smoothly.
- The fix was adding 2 ~~1000μF~~ 100μF capacitors between GND and 7V input.

EXTRA WIRES
- As I was putting together my box for the final time, I found a wire underneath it where an opening is. This made me panic because both ends were disconnected so finding its original place would be next to impossible without undoing the box due to its limited size. After a long inspection and various testing (exhaustive) I determined the wire was there before me. I was relieved but also disappointed I had spent an hour trying to solve this riddle.

OVER SUPPLYING Components
- power supply accidentally supplied 30V to all of my circuit, where 7V was maximum.
- Needed to replace PCA 9685 and Megaboard due to shorted circuit.
- TIP: Dont!!! be careful with ~~fragile~~ components.
- TIP: A circuit is shorted when it draws the maximum current provided. USE a multimeter to check for the proper working of your components.

STEPPER TORQUE
- When the weight of the arm was too displaced relative to the box horizontally, the stepper would not turn.
- An "back of the envelope" calculation provided a simple equation to the perpendicular distance the center of mass had to be from the box. This was easily implemented in code by inputting an x,y value with a boolean an output

CIRCUIT



ESP 32 CAMERA

a low cost development board with WIFI camera. It allows creating IP camera projects for video streaming with different resolutions.

In this project, I used it's facial recognition feature, to locate the face center so that I could calculate the pan and tilt the arm would have to move to center the face.

It requires 5V (although its data sheet specifies 3.3V) and communicates with the Megaboard via serial pins.

This communication was initially a source of problem as all resources using the ESP 32 CAM didnt use a Megaboard. Looking through Arduino's serial pins and initializing the Serial monitor within the code resolved the issue.

PCA9685

This servo driver uses I2C interface to communicate with up to 16 servo ~~de~~ motors (with 1 board) and up to 96 (with multiple boards). It only requires 2 SDA and SCL pins, freeing up space for other pins.

The advantage of this driver besides the reduction of pin uses, are a seperate power spins for the servo motors. This allows to supply the motors with a higher voltage and current than with 5V supplied by the board.

STEPPER MOTOR

the stepper motor used was the same one provided by the kit. Its torque should have been sufficient assuming the only force to overcome was ~~friction~~, the inertia of the arm. However due to the extension of the arm, this caused the base plate to have increased friction, reducing the effectiveness of the motor.

IR RECIEVER

The IR Reciever allows for the transmission of hexadecimal 1 byte ~~values~~ values through infrared. A limitation of this device is the range of the remote limited to a 5meter. Another limitation is that one can only transmit predefined values (out of 21 buttons).

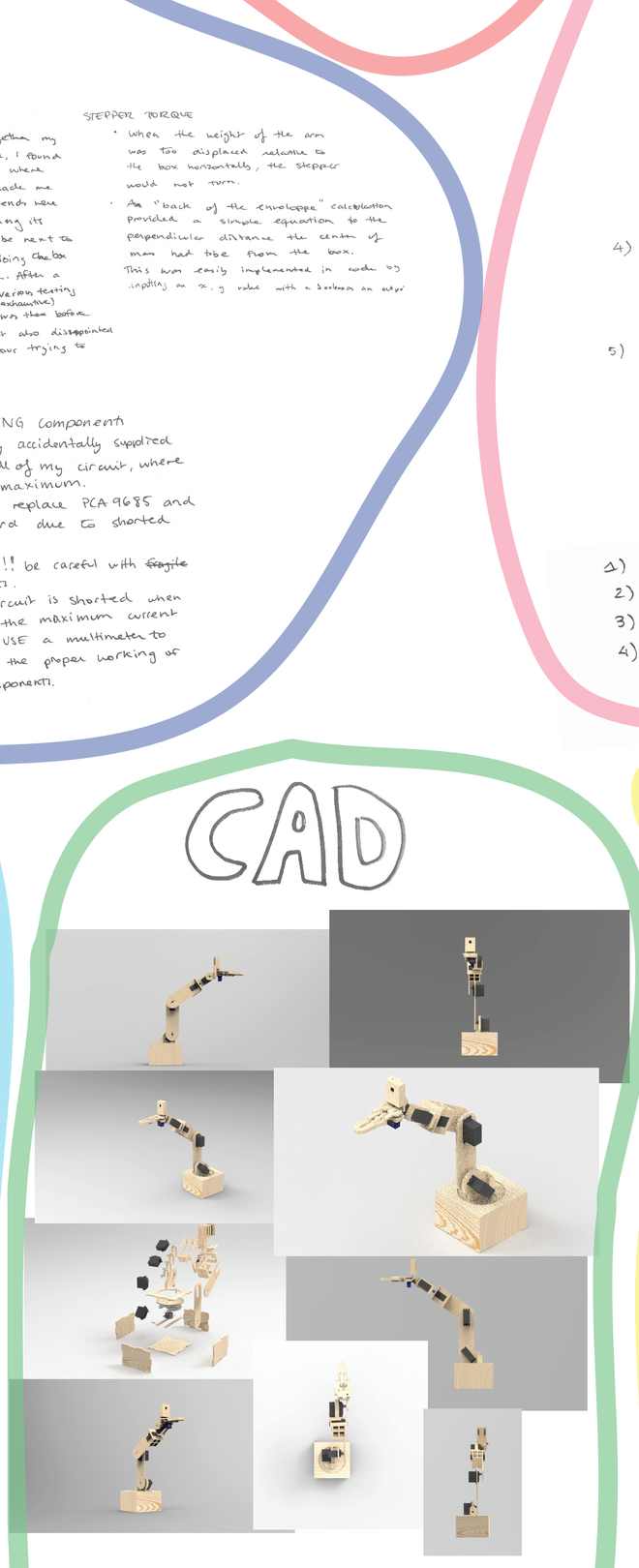
POWER SUPPLY

ORIGINALLY, I had bought a variable wall plug (wall wart) that supplied up to 12V and 2A. However upon testing via the multimeter, when on the setting of 7.5V (perfect for my ~~subsystem~~ circuit) the plug outputted 8.2V. This could have had disastrous consequences for my circuit, burning it out. So I decided to stick with the TENMA 72-2540 DC power supply a reliable source.

SERVOS MG996R

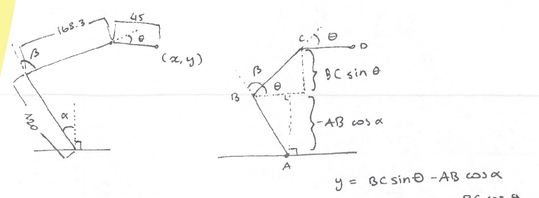
These digital servos contain all metal gearing which result in 10Kg/torque and 55g in weight. They were chosen for their price to quality ratio. as they come in a pack of 4. the servo for the claw ~~was~~ was SG90. this servo did not provide enough torque to hold anything impressive.

CAD



INVERSE KINEMATICS

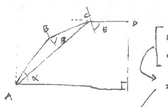
- one of the hardest parts of this project was finding the angles of the servos so that the camera would be at a certain X,Y position.



$y = BC\sin\theta - AB\cos\alpha$

$x = -AB\sin\alpha + BC\cos\theta$

$\beta + \theta = \frac{\pi}{2} + \alpha$



$Cx = Dx - CD$

$Cy = Dy$

$AC = \sqrt{(Cx)^2 + (Cy)^2}$

$\frac{\pi}{2} + \beta = \cos^{-1}\left(\frac{AB^2 + BC^2 - AC^2}{2 \cdot AB \cdot BC}\right)$

β → found

$\alpha + \left(\frac{\pi}{2} + \beta\right) + \left(\frac{\pi}{2} + \theta\right) + \frac{\pi}{2} + \frac{\pi}{2} = 3\pi$ (sum of pentagon angles sum to 3π)

$\alpha + \theta = \pi - \beta$

α → found

$\alpha = \cos^{-1}\left(\frac{AC^2 + AB^2 - BC^2}{2 \cdot AC \cdot AB}\right) + \tan^{-1}\left(\frac{Cy}{Cx}\right)$

θ → found

$\theta = \pi - \beta - \alpha$