Head-Controlled Mouse

ECE 445 Design Document

Team 44

Amanda Favila (afavila2)

Asher Mai (hanlinm2)

Lauren Wilcox (lwilcox4)

TA: Sainath Barbhai

Professor: Victor Gruev

Senior Design Spring 2023

23 February 2023

1. Introduction

1.1. Problem

Using a head-controlled mouse can benefit people who want to move their mouse faster, switch between typing and clicking without taking their hands off the keyboard, or have disabilities that make using a standard mouse difficult. According to the CDC, 24% of adults in the United States have arthritis, which is the leading cause of work disability [2]. Teamstage asserts that "about 50% of jobs need employees to have technology skills, but it's estimated that by 2030, 75% will have such requirements" [3]. Because so much work uses computers these days, there is a need for computer accessibility products for people with such disabilities to utilize for both entertainment and work purposes. On top of arthritis, there are many other conditions or disabilities that inhibit arm and hand movement and interfere with computer use such as carpal tunnel, partial paralysis due to spinal cord injury, Parkinson's disease, Huntington's disease, amyotrophic lateral sclerosis, cerebral palsy, and stroke-related motor cortex damage, as listed by Frontiers [4].

Although eye and head-controlled mice have been invented to curb these issues, these products typically use headbands, require an expensive camera setup, and the programs are not universal to any device from Mac to PC to iPad. Some examples of such products are Smyle mouse [5] and Mouseware [6]. Headbands may not be the most comfortable solution for everyone, so our product will not use one. Many people dislike using cameras due to privacy concerns, so the camera component of this solution can be eliminated by substituting accelerometers and gyroscopes. There is also a way to make this technology more universally accessible to all devices, via a USB wireless receiver. All three of these changes are what will set our product apart from similar ones on the market.

1.2. Solution

Our solution is to create a device that will process the user's head motions to control the cursor on whichever device they are using. This device will be attached to a hat, which is more comfortable for the user than the typical headband, is quicker and more convenient to put on and take off, and can balance the weight of the device and its battery more evenly. After calibration, this device will track when the user turns their head up, down, left, and right to move the cursor on their screen accordingly, and then use a specific head tilt to click.

More specifically, the head motions will be tracked on the hat using a combination of gyroscopes and accelerometers. The hat will be powered with a standard battery pack, which will be separate from the hat and attached via wires in order to reduce the weight of the hat. The hat will also contain a microcontroller that processes the head movements and will send signals to the receiver subsystem connected to the desktop/laptop devices which will then move the mouse accordingly. We will utilize a voltage regulator as well on our PCB to step the voltage down depending on the needs of our components.

1.3. Visual Aid

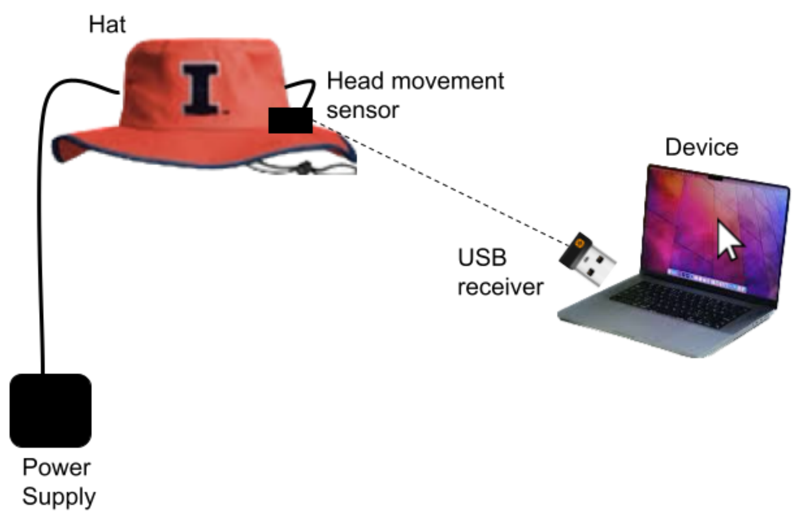

Figure 1: Visual Aid of Head-Controlled Mouse device

1.4. High-Level Requirements

- The device must have a successful calibration sequence that calculates appropriate distances and speeds for the cursor to move based on the user's specific head movements.
- The device must be able to accurately move and click the mouse cursor based on the user's head movements. This means that when the user moves their head up, down, left, and right, the mouse cursor will move up, down, left, and right, respectively.
- The device must be able to be used on both Macs and PCs.
- The device must utilize user adjustable sensitivity that can map different cursor speed to the same head rotation speed.

2. Design

2.1. Block Diagram

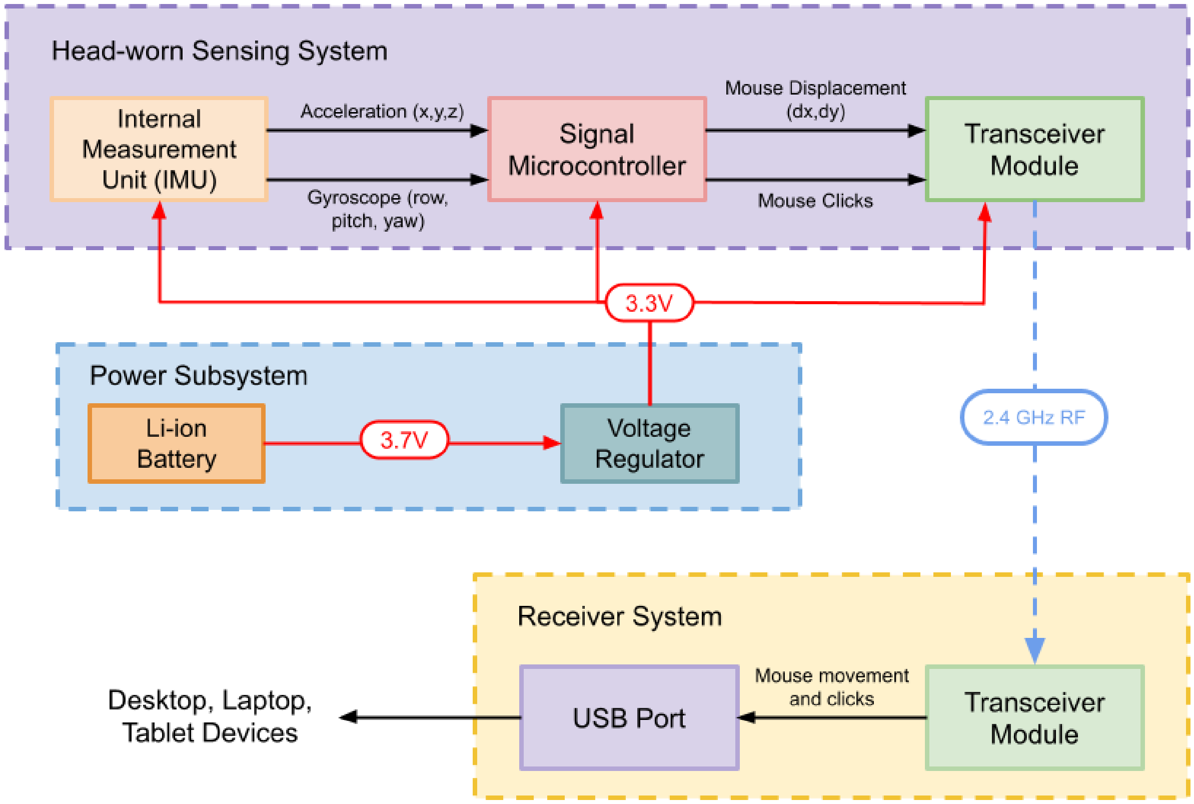

Figure 2: Block Diagram

2.2. Physical Design

All of the components of the head-worn sensing system as seen in the purple section in Figure 2 will be enclosed in a 3-D printed box. This box will have a snap on lid and a hole for the power supply wire to connect to the IMU-Microcontroller-Tranceiver system. This enclosure will be attached to the hat, and the power supply can be placed in the user's pocket or on a nearby surface.

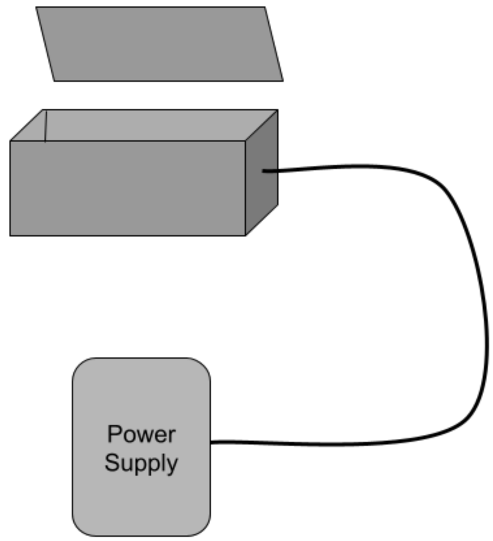

Figure 3: Sensor Enclosure

2.3. Internal Measurement Unit (IMU)

An IMU is needed to measure the head rotations of the user. This will be mounted on our PCB which will be placed in the box enclosure as seen in Figure 3. The IMU is an integral part of our design as it contains the gyroscope and accelerometer which will generate the data of the direction of the user's head movement and the speed in which the user's head moves. The acceleration and gyroscope raw data will then be sent to the signal microcontroller, which will be processed into data that maps to mouse cursor displacement. This newly converted data will be then sent through to the transceiver to get to the computer. The IMU will receive its required supply voltage of 1.8V - 3.6V from the power supply subsystem. No calibration curve was given in the design document, so we will calibrate the data with the python library **imucal** using the Ferraris calibration method that is applied to 6 degrees of freedom IMUs. Our ideal data set for this calibration will be the data generated from the user moving their head from looking straight (yaw = 0°) to the left (yaw = -90°), and the same with right, down, up, and the head tilts.

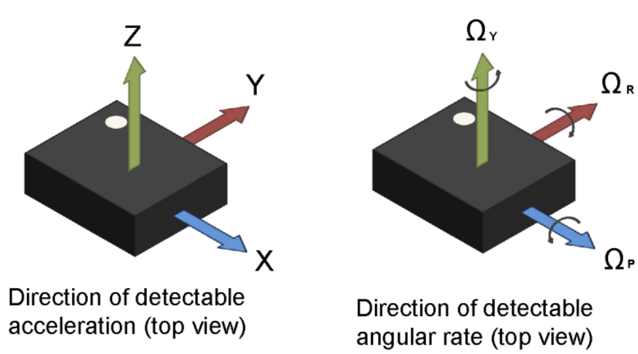

Figure 4: IMU Directions

IMU Requirements	**IMU Verifications**
The IMU should be able to detect the differences in orientation based on the user's head movement, i.e. movement up, down, left, right, and head tilts.	We will display the data measurements generated by the IMU through an Arduino to verify that correct directions are being calculated.
The IMU should be calibrated to convert a 90° range of yaw movement to 100% width of any screens, and convert a 50° range of pitch movement to the 100% height of any screens.	We will verify that the mouse cursor can be moved from one side of the screen to the other without moving the head further than the degree constraints. This can be further verified by moving only 45° range of yaw and ensuring the cursor only moves 50% across the width of the screen.
The IMU should be able to measure the speed of the user’s head movements.	We will use an Arduino to display the angular velocity data and confirm that the values are correct relative to the user’s head movements.
The IMU should have minimal latency in sending data, less than 0.02 second.	When mapping the IMU data to the mouse cursor on a computer, we will time the difference between the user moving their head and the cursor moving on screen. We are also electing to use SPI protocol rather than I2C, which is faster since it is a 4 wire protocol where data can be sent and received at the same time.

2.4. Microcontroller

The microcontroller is a programmable chip that will carry out our desired functions of mapping the gyroscope and accelerometer data from the IMU to the desired mouse position on the screen. The microcontroller will take in the following data from the IMU: acceleration in the x, y, and z directions, and gyroscope data of row, pitch, and yaw. After processing the data according to our code, the microcontroller will output the transformed data in the form of mouse displacement as well as clicks, and these outputs will go to the transceiver to then be sent to the receiver.

Microcontroller Requirements	Microcontroller Verifications
The microcontroller must have enough	We can verify that the microcontroller that

input and output pins for our data. Input: ● Acceleration X ● Acceleration Y ● Acceleration Z ● Gyroscope row ● Gyroscope pitch ● Gyroscope yaw Output: ● Mouse displacement X ● Mouse displacement Y ● Left click ● Right click ● Scroll	we chose will have the correct number of I/O pins by consulting the datasheet. The datasheet states that there are 23 programmable I/O lines, which is enough for our data even using 4-wire SPI protocol.
We must be able to use a microcontroller on a breadboard as part of our prototype as well as on a PCB as part of our final design.	We have solved this requirement by actually using 2 microcontrollers. The one we have selected in this document will be part of our final design, soldered into the breadboard. For prototyping, we will use the microcontroller from an Arduino on a breadboard. It has the following datasheet: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-metro-mini.pdf The pinouts of these 2 microcontrollers do not exactly match up, however we can just wire the prototype and the final version differently. Our main concern is just being able to tweak the code and mess around with the prototype, and since we will not be using any complicated loops in our program (it will be pretty simple math), it will translate well between the two microcontrollers.
The microcontroller must communicate in SPI standard so that it can communicate with our chosen IMU.	This is verified in the datasheet for the microcontroller we selected.

2.5. Transceiver

We will use 2 transceivers for our design. The first transceiver's purpose is to receive the output of the microcontroller data, which includes both the position of the mouse as well as whether or not the mouse is clicked (left clicks, right clicks, and potentially scrolling). The second transceiver will plug into the computer's USB port and it will listen for the data from the first transceiver.

Transceiver Requirements	Transceiver Verifications
The transceivers must be able to communicate with the receiver at the same frequency, 2.4 GHz RF.	This is verified via the datasheet. We can also use an oscilloscope and send a single pulse in the frequency domain to measure at which frequency this pulse is sent at.

2.6. Power

The power supply is what gives the potential difference to the circuit so that current can run through and all of the components have power to operate together. For our power supply, we chose to use a rechargeable battery so that the device can be recharged and reused without the user needing to keep purchasing batteries. We have selected parts for both the batteries themselves as well as the USB-C charging port that will be used to recharge the batteries. The IMU we have chosen requires a supply voltage of 1.8 V (a range of 1.71 V - 3.6 V) as well as a high-performance current of 0.55 mA. The microcontroller we have chosen requires a supply voltage of 2.2 V - 5 V as well as a current of 13 microAmps to 5 mA, depending on the frequency of oscillation and the supply voltage. Because of all of these requirements, we have selected our power supply to output a constant 3.7 V +/- 1.3 V.

Power Supply Requirements	Power Supply Verifications
Supply a voltage of 3.7 V +/- 1.3 V.	We can verify that the power supply will output the desired voltage by measuring the voltage across the two terminals of the battery using a digital multimeter. We can verify that the power supply will output the desired currents to each of our components by using a shunt resistor and measuring the voltage across the resistor and using Ohm's law to determine the current.
Input/Output protection: What would happen if a user inserted a plug or battery backwards?	We can prevent users from plugging the battery in backwards by being very intentional with our wire colors, labeling, and instructions. However, if this were to happen, we will be sure to place a diode in the circuit, so that if the power supply is plugged in the incorrect way, no current will flow through the diode and we will have an open circuit.

Environment: Will your project function in hot or cold conditions?	According to the datasheet, the operating temperature of the power supply is anywhere from 0 degrees Celsius to 50 degrees celsius, while still maintaining up to a 4.1 V charge. This verifies that our power supply will function in all typical room temperatures, and even in pretty cold or pretty hot temperatures.

2.7. Tolerance Analysis

One aspect of the design that poses a risk to the successful completion of the project is the accuracy of the cursor movements compared to the head movements of the specific user. We plan to mitigate this issue using a calibration function during the setup of our device. Note the axes in figure 3 below, where the sensor on the user's head will act as the origin, and they face the screen along the roll axis. The bounds of these definitions are:

- $-180° \leq roll \leq 180°$
- $-90° \leq pitch \leq 90°$
- $-180° \leq yaw \leq 180°$

We can generally define the user functions as follows:

- Rest (mouse stays idle): roll = 0°, pitch = 0, yaw = 0°
- Up (mouse moves up): roll = 0°, 0° < pitch ≤ 90°, yaw = 0°
- Down (mouse moves down): roll = 0°, -90° ≤ pitch < 0°, yaw = 0°
- Left (mouse moves left): roll = 0°, pitch = 0°, -180° ≤ yaw < 0°
- Right (mouse moves right): roll = 0°, pitch = 0°, 0° ≤ yaw ≤ 180°
- Tilt left (left click): -180° ≤ roll < 0°, pitch = 0°, yaw = 0°
- Tilt right (right click): 0° ≤ roll ≤ 180°, pitch = 0°, yaw = 0°

Note that we will have to alter these definitions of positive and negative directions according to the parameters of the specific gyroscope that we use.

We will have to account for noise in this situation. It is unrealistic to expect the user to sit completely still all of the time. Therefore, we can implement a sort of high pass filter that will only allow intentional movements to control the mouse. The goal of this filter can be to disregard any movements that are less than about 5°. This number will change after testing the device out, and may also change based on the calibration of the specific individual.

Our calibration will work by prompting the user to give samples when they are facing forward at rest, as well as when they are looking up, down, left, right, tilting their head left, and tilting their head right. The data taken from the user at rest will be useful in filtering out any additional noise specific to the user's movement. All of the calibration measurements will give our program a good baseline for interpreting the user's movements since people have different abilities. After taking a calibration of the user, the above definitions of motion will become more specific based on the user's specific range of motion.

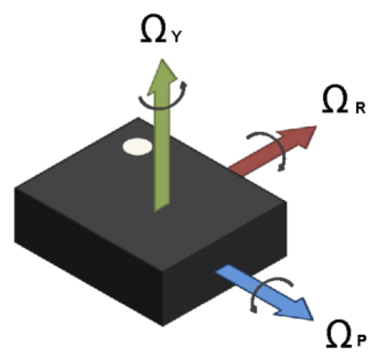

Figure 5: Illustration of roll, pitch, and yaw of gyroscopic motion

The datasheet for the IMU we selected talks about a feature it has called Significant Motion Detection. This function generates an interrupt when a ‘significant motion’ that could be due to a change in user location is detected. So, some filtering is already done inside the IMU to not take into account every microscopic motion going on. However, we will likely have to add our own high-pass filter in the software on top of that. During our prototyping, we can experiment with different resistor and capacitor values until we are satisfied.

Another quantification worth considering is how we would like to map the degrees of physical head rotation to the amount of pixels moved by the mouse. This will obviously take some testing to get smooth, but we can consider some approximations to begin with. To start, we can think about the degrees of head rotation that are comfortable for a user. We want the user to be able to keep their eyes on the screen while rotating their head so that they can see what is going on and how the mouse is responding to their movements. This means that we should not expect the user to rotate their head 90 degrees to the left and right for significant movement. A comfortable range of motion could be 45 degrees to the left, right, up, and down. From there, we need an estimation of how many pixels we want to move for each degree of rotation. An industry-standard full HD 1080p monitor has a resolution of 1920 pixels x 1080 pixels. A reasonable rate would be 15 pixels per degree of movement, which would allow the user to traverse the entire screen in a matter of seconds by fully moving their head 45 degrees, while also allowing the user to make precise mouse movements with smaller head movements.

3. Cost & Schedule

3.1. Cost Analysis

Cost of labor: $30/hr ✕ 12 hrs/week ✕ 10 weeks ✕ 3 members = $10,800

Description	Manufacturer	Quantity	Ext. Price	Links
Baseball cap		1	$10.00	
LDK120M33R (Voltage Regulator)	STMicroelectronics	1	$0.91	Link
LSM6DSOTR (Inertial Measurement Unit)	STMicroelectronics	2	$6.60	Link
ATMEGA328P-AU (Microcontroller)	Microchip Technology	1	$2.86	Link
BLUENRG-MSQTR (Transceiver)	STMicroelectronics	1	$4.72	Link

LP523450JU+PCM+2 WIRES 70MM (Battery)	Jauch Quartz	1	$11.30	Link
USB4105-GF-A (USB-C charging port)	GCT	1	$0.81	Link
74LVC1T45GW,125 (Level Shifter)	Nexperia USA Inc.	1	$0.46	Link

Total Cost of Project: $42.98 (Parts) + $10,800 (Labor) = $10,842.98

Additional components needed to hook up all of our parts:

100 nF filter capacitors for the IMU		2		
22 kOhm resistor for the transceivers (1 each)		2		

3.2. Schedule

Week	Deliverables	Division
2/13	Finish the Design Document and Team Contract.	All
2/20	Purchase all parts listed in the Design Document as well as anything needed for prototyping.	All
	Prepare for the Design Review.	All
	Draft schematic and PCB.	Lauren
	Learn how to use and communicate with the wireless USB receiver.	Asher
	Start planning how data processing will be done (converting accelerometer and gyroscope data to mouse location).	Amanda
2/27	Design Review and PCB review with Instructor and TAs. Purchase all needed components.	All
	Make any necessary changes to the PCB.	Lauren
	Set up a program project so we can start writing code.	Asher
	Plan how the sensors will be securely attached to the hat.	Amanda
	Pass PCB audit and place order by Tuesday 3/7.	All

3/6	Complete Teamwork Evaluation I by Wednesday 3/8.	All
	Assemble breadboard prototype. Begin testing components to get comfortable with how they work.	Lauren
	Establish a wireless connection between the hat and the USB receiver. Begin writing any code necessary for communication.	Asher
	Begin writing data processing code.	Amanda
3/13	Spring Break :)	All
3/20	Assemble PCB. Test connections.	Lauren
	Finalize code.	Asher and Amanda
3/27	Pass second round PCB audit and place order by Tuesday 3/28.	All
	Complete individual progress reports by Wednesday 3/29.	All
	3D model and print an enclosure for the power supply.	Asher
	Connect the power supply to the project with appropriately selected and sized wires.	Lauren
	Attach sensors to the hat.	Amanda
	Begin testing.	All
4/3	Continue testing, fixing bugs, and refining project.	All
4/10	Complete Team Contract Fulfillment by 4/14.	All
	Prepare for Mock Demo.	All
4/17	Give Mock Demo during the weekly TA meeting on Tuesday 4/18.	All
	Prepare for Final Demo and Mock Presentation.	All
4/24	Give Final Demo to instructor and TAs.	All
	Give Mock Presentation with TAs. Prepare for Final Presentation.	All
	Begin Final Paper.	All
5/1	Give Final Presentation to instructor and TAs.	All
	Complete final paper by Wednesday 5/3.	All
	Turn in Lab Notebooks by Thursday 5/4.	All
	Complete Lab checkout with TA and attend award ceremony on Thursday 5/4.	All

4. Ethics & Safety

4.1. Ethics

Our team carefully took the IEEE Code of Ethics into consideration when developing our head-controlled sensor.[1] The safety of the users is important so we had to consider the different head movements that would cause the cursor to move so that they would not cause headaches or chronic discomfort. This also includes using safe batteries in enclosures that will not incur any burns and a well-balanced weight distribution of the device on the hat.

We also work in a peer and faculty-reviewed system to ensure that our technical work is legitimate and safe with realistic claims, as stated in section 5 of the IEEE code.[1] Each part of our project will be looked over by all people on our team and we will be open to criticisms and suggestions for improvement. Our team has also undertaken all relevant CAD, lab safety, and other technical training needed to keep everyone safe when working on the project and to be in compliance with campus policies in the labs. This way we will ensure that our technical competence is at a level reliable to be successful in our project as mandated by section 6 of the IEEE code.[1]

The ethical implications of our project are something we thought of seriously when developing the concept of our device, and it was a big factor in implementing a camera-free design. This will mitigate privacy concerns with being on camera and where that data may go. By only tracking the data of the head's direction which will be processed into a mouse displacement, we will avoid ethical breaches in any personal data.

4.2. Safety

To maintain the safety of our users, we follow the safety procedures outlined by all individual parts and are choosing to put the part of the device worn by the user in an enclosure to avoid any potential contact with the parts. The battery we are using for our power supply will be rechargeable to avoid having to replace them, and we will have it enclosed with a diode connected to the circuit to cause an open circuit if the battery was plugged in backwards. We will additionally instruct users to adjust the hat to properly fit their head to promote proper weight support for the device and avoid any discomfort.

References

[1] IEEE code of ethics. IEEE Policies, Section 7 - Professional Activities.
https://www.ieee.org/about/corporate/governance/p7-8.html

[2] CDC. National Center for Chronic Disease Prevention and Health Promotion (NCCDPHP).
https://www.google.com/url?q=https://www.cdc.gov/chronicdisease/resources/publications/factsheets/arthritis.htm%23:~:text%3DIn%2520the%2520United%2520States%252C%252024,%252C%2520rheumatoid%2520arthritis%252C%2520and%2520lupu&sa=D&source=docs&ust=1675989117958870&usg=AOvVaw1pRn72DphIGMQosuPT3271

[3] Teamstage. Technology in the Workplace Statistics for 2022: Is Tech Helping or Hurting Us?.
https://teamstage.io/technology-in-the-workplace-statistics/#:~:text=Today%2C%20about%2050%25%20of%20jobs,75%25%20will%20have%20such%20requirements

[4] Frontiers. Arm and hand movement: current knowledge and future perspective.
https://www.frontiersin.org/articles/10.3389/fneur.2015.00019/full#:~:text=Neurological%20conditions%20that%20can%20affect,cord%20injury%20at%20cervical%20levels

[5] Smyle Mouse. Head Mouse for Hands-free Computer Control.
https://smylemouse.com/

[6] DextroWare. A Revolutionary Head controlled mouse called "Mouseware".
https://dextrowaredevices.com/head-controlled-mouse/