UNIVERSITY OF CALIFORNIA,
IRVINE

Headmouse: A Simple Cursor Controller based on Optical Measurement of Head Tilt

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Engineering

by

Ali HeydariGorji

Thesis Committee:
Professor Pai H. Chou, Chair
Professor Nader Bagherzadeh
Associate Professor Mohammad Al Faruque

2018

DEDICATION

I dedicate this thesis to my family, my beloved parents and sister whom I haven't seen for a long time, but they have always been there to support me, motivates me, and give me sincere advice. Undeniably, I owe all my academic achievements to them, and this dedication is the smallest gratitude that I could express to them.

TABLE OF CONTENTS

Page

LIST OF FIGURES **v**

LIST OF ALGORITHMS **vi**

ACKNOWLEDGMENTS **vii**

ABSTRACT OF THESIS **viii**

1 Introduction **1**
1.1 Pointing Devices and Assistive Technology 1
1.2 Head Tracking 2
1.3 Contributions 3
1.4 Thesis Outline 4

2 Related Work **5**
2.1 Interaction Methods 5
2.1.1 Camera-based systems 6
2.1.2 Tongue trackers 6
2.1.3 EEG-based systems 6
2.1.4 Voice recognition 7
2.2 Head Tracking 7
2.2.1 Inertial Sensing 7
2.2.2 Muscle tension 7
2.2.3 Optical reflection 8
2.3 Summary 8

3 System Design **10**
3.1 Technical Approach 10
3.1.1 Reflective IR Sensors 10
3.1.2 Signal Conditioning 12
3.2 Modes of Operation 13
3.2.1 Joystick Mode 13
3.2.2 Direct-mapping Mode 14
3.3 Click function 16
3.3.1 Capacitive Touch Sensor 16

3.3.2 Force-Sensing Resistors 16

4 Implementation **18**
4.1 MCU and Wireless Communication 18
4.2 Optical Sensor 19
4.2.1 IR Transmitter-Receiver Pairs 20
4.2.2 PWM Control and ADC 21
4.3 FSR buttons for clicking 22
4.4 Power Subsystem 22
4.5 Mechanical Design 23
4.6 Data Processing 24
4.6.1 On-Host Processing 24
4.6.2 On-Node Processing 24

5 Results and Discussion **26**
5.1 Experimental Setup 26
5.2 Performance Metrics 27
5.3 Results 29

6 Conclusions **33**

A Schematic of Ecomini **35**

Bibliography **37**

LIST OF FIGURES

Page

1.1 Example of pointing devices. (a) Mouse. (b) Track ball. (c) Track pad. (d) Touch screen. 3

2.1 Examples of methods to control cursor position.[10, 12, 13, 3] 9

3.1 Location of the optical sensors and the CC2541 module including the BLE, ADC, and the battery. A US quarter coin is placed for size comparison but is not part of the system 11
3.2 Signal changes during the pitch/ yaw movement. 12
3.3 FSR sensors on the chick to detect clicking. 17

4.1 The system block diagram. 19
4.2 Actual circuit of the sensor board. 20
4.3 Source code for reading data from sensors. 21
4.4 Sensor circuit to detect left/right click. 22
4.5 Main deck design. 23
4.6 Source code for reading data from sensors. 25

5.1 The testing platforms, (a) User tries to follow a predefined path in the joystick mode. (b) User tries to guide the cursor into a target in the joystick mode. (c) Following the path in direct mapping mode. (d) Guiding the cursor to the target in direct mapping mode. 27
5.2 Demonstrating Fitts's law. 28
5.3 Path efficiency with respect to index of difficulty for both modes. 30
5.4 Example of same measures with different index of difficulty. 31
5.5 Throughput with respect to index of difficulty for both modes. 32

6.1 Headmouse device. 34

A.1 Ecomini, uses as the main part of the system to collect signals and send it through BLE. 36

LIST OF ALGORITHMS

		Page
1	Joystick calibration: Adjusting the thresholds.	13
2	Joystick mode.	14
3	Direct mapping mode, calibration phase.	15

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Prof. Pai H. Chou for his consistent guidance from the beginning of this project up to its last moments. His unwavering assistance, indispensable advice, and technical instructions showed me the proper direction to excel in this project. My sincere gratitude goes to him because he devotedly cooperated in our publication.

My appreciation also extends to committee members, Prof. Bagherzadeh and Prof. Al Faruque not only because of their priceless time and review but also because I learned many fundamentals in their courses which directly or indirectly contributes to this project.

Finally, I'm grateful to my colleague at University of California Irvine (UCI) and National Tsing Hua University (NTHU) who helped me during the developing process including Mahya Safavi, Hsin-Chung (Andrew) Chen, James Chan, Zhe-Ting Liu, and Yu-Hung Yeh.

ABSTRACT OF THE THESIS

Headmouse: A Simple Cursor Controller based on
Optical Measurement of Head Tilt

By

Ali HeydariGorji

Master of Science in Computer Engineering

University of California, Irvine, 2018

Professor Pai H. Chou, Chair

The ability to point using a mouse or a touch screen is an indispensible part of modern human-computer interactions. However, people with motor impairment need assistive technologies. Existing technologies based on voice command, eye tracker, head tracker, tongue tracking, muscle-tension tracking tend to be complex or expensive. We propose a new system that uses head position to control the cursor position using a very simple, low-complexity infrared-based mechanism. It uses two infrared (IR) LEDs to measure the distance between the chin and the sensor deck. Simple algorithms convert measured data to X and Y coordinate on the screen. The Device sends either raw or processed data to host system through Bluetooth Low Energy (BLE) technology. Experimental results show the proposed head mouse to be easy and intuitive to use while consuming low power and is easy to build.

Chapter 1

Introduction

Pointing on a two-dimensional display has been a fundamental operation in graphical user interface in the past three decades and will continue to remain important in the foreseeable future. Pointing devices have evolved from desktop mice and trackballs to trackpads and touch screens. Fig. 1.1 shows some examples of pointing devices.These new inventions provide users with more degree of freedom and better interaction with systems. In the case of touch screen, there is no need to drag the pointer around. Instead, the user can directly tap where the pointer needs to be. This reduces the stress over muscles and enhances user experience. Unfortunately, not all users can access conventional pointing technologies equally well, and assistive technologies have been developed. We propose our assistive technology based on head tilting.

1.1 Pointing Devices and Assistive Technology

A major assumption with all pointing devices is that the user has (at least) one free hand, but not all users have the ability to use their hands to interact with machines. In the United States alone, more than 3 million people have a disability in their hands or forearms, including paralysis, orthopedic

impairments, either congenital or injury related, and this number is expected to double by 2050. Approximately 185,000 amputations occur in the US every year [2, 7]. For patients suffering from amputation or paralysis, alternative mechanisms must be used to enable their interaction with computers. These mechanism targets other parts of the body that can signify commands in a way that can be discerned by sensors. For example, small movements in fingers, eyes, tongue, head, or other muscles can be interpreted as distinct commands. However, these commands should be defined in a way so that natural body movements can be easily distinguished and filtered.

1.2 Head Tracking

One of the best alternative to hand gesturing is head gesturing. Many who have lost motor control of the hand can still control their head using their neck muscles. They are likely to have control of their eyes, too, but we consider the head for the greater range while freeing up the eyes for the more critical input. By mapping head moves to the location of the cursor on the screen, we can construct a new pointing device that does not require limb control, has several degrees of freedom, and enables fine control of the pointer's location.

Ways to detect head movements include inertial sensing, muscle tension, and optical reflection. Such a device should be able to pick movement commands fast enough to support real-time inteaction, as delays larger than a tenth of second can diminish the user experience. It also needs to be wearable, which requires the design to be compact, lightweight and standalone. That means the system should not include many wires and pieces that make it awkward to wear. More importantly, since battery is often the most bulky component in wearable devices, the system must be power-efficient to keep the battery size small.

(a) (b)

(c) (d)

Figure 1.1: Example of pointing devices. (a) Mouse. (b) Track ball. (c) Track pad. (d) Touch screen.

1.3 Contributions

All these considerations have led us to a new kind of mouse that is easy to control by held tilt, power efficient, accurate, and compatible with modern personal computer systems. Our proposed device is to be mounted on the collar bone area to track the head movement for cursor control. It considered as a non-intrusive device since there is no need to contact with internal organs. The mouse button could be implemented as an integral part of head tracking, but we chose to evaluate the option of using the tongue or cheek to signify pressing the mouse button. It can be integrated with very low material cost and minimal additional firmware.

1.4 Thesis Outline

The rest of this thesis is organized as follows. Chapter. 2 surveys related work on assistive pointing devices. Chapter. 3 presents our technical approach including the two modes of operation and clicking function. Chapter. 4 describes our implementation based on an integrated RF-enabled microcontroller, optical sensor-actuator units, and force-sensing resistors. Chapter. **??** evaluates our system with an analysis of the results collected from actual users. Finally, Chapter. 6 concludes this thesis with directions for future work.

Chapter 2

Related Work

In recent years, many assistive technologies have been developed for enabling hands-free cursor control. These techniques target users with a wide range of motor disabilities, and they all have been proven to be effective in their own specific ways. However, they all come at a relatively high cost. This chapter first classifies them by interacting methods, followed by related work that rely on head control.

2.1 Interaction Methods

Assistive mice that rely on different parts of the body for control have been proposed. They can be divided into eye trackers [5], head trackers, tongue trackers [6], brainwave (EEG) sensors [9], and muscle tension sensors (EMG) [14].

2.1.1 Camera-based systems

This method relies on a stream of images of the user to detect movements of the body, head, or eyes. Tracking changes between successive frames, require intensive image processing, which consumes nontrivial computational power and thus may not last long on a compact battery. This makes them unsuitable for portable long-term usage on the order of weeks or months [1].

2.1.2 Tongue trackers

Tongue trackers can sense tongue movements and convert them to movement commands. They usually require special sensors to be placed in the mouth. Aside from likely uncomfortable user experience, it can cause hygiene concerns, especially for energy storage. Being in contact with saliva, all parts need to be waterproofed and kept in sealed packages. Note that mouth-mounted devices may be considered intrusive if not invasive.

2.1.3 EEG-based systems

Electroencephalography (EEG) is an electrophysiological monitoring method to measure the electrical activity of the brain. It measures voltage fluctuations resulting from ionic current within the neurons of the brain. The features extracted from EEG signals can direct a pointer on the screen. However, it requires complex algorithms and computations to extract the features. Although most EEG systems are considered noninvasive, they require a large number of electrodes and wires to be mounted on the scalp. Therefore, they are complex and can be uncomfortable to wear. Moreover, they are prone to noise issues, and the performance may be poor in terms of speed and accuracy [8].

2.1.4 Voice recognition

Voice-based systems can handle text input for verbal commands and dictation. In general, voice commands have a low degree of freedom since each individual command needs to be spoken. In addition, the computational load due to voice processing can consume high power and and can incur long latency.

2.2 Head Tracking

Assistive mice that use head tracking can be further divided based on their sensing modalities.

2.2.1 Inertial Sensing

Inertial sensors, including gyroscopes and accelerometers, can provide the data for determining the location and angles of the head. The overall system can be simple and relatively cheap, as inertial sensors often incorporate accelerometers and gyroscopes in a single miniature package. However, the output data is prone to heavy noise and bias miscalibration.

2.2.2 Muscle tension

Muscle-tension sensors can be based on Electromyography (EMG) or photoplethysmogram (PPG). EMG records the electrical activity produced by skeletal muscles . It detects the electric potential generated by muscle cells when these cells are electrically or neurologically activated. PPG, on the other hand, is an optically obtained plethysmogram, a volumetric measurement of an organ. A PPG is often obtained by using a pulse oximeter that illuminates the skin and measures changes in light absorption. Both can be used to detect the head position and movements by measuring

muscle tension in the neck area. However, EMG can be intrusive as it requires skin preparation or use spiky electrodes that can cause discomfort, while PPGs often suffer from noise and low accuracy.

2.2.3 Optical reflection

Optical reflection can be applied to head position detection. It entails emitting an optical signal towards part of the head such as the chin and measure its reflection. The distance or motion can be measured by changes in the amplitude or phase of the reflected optical signal. Few previous works have studied this method. Although it can suffer from ambient light interference, can several ways can eliminate such noises. Our proposed systems relies on optical reflection to detect head movements.

2.3 Summary

This chapter has surveyed assistive technologies for pointing devices, including different interaction methods and sensing modalities. Next chapter will explain how our proposed method detects the head position and maps the moves and positions to the cursor location on the screen.

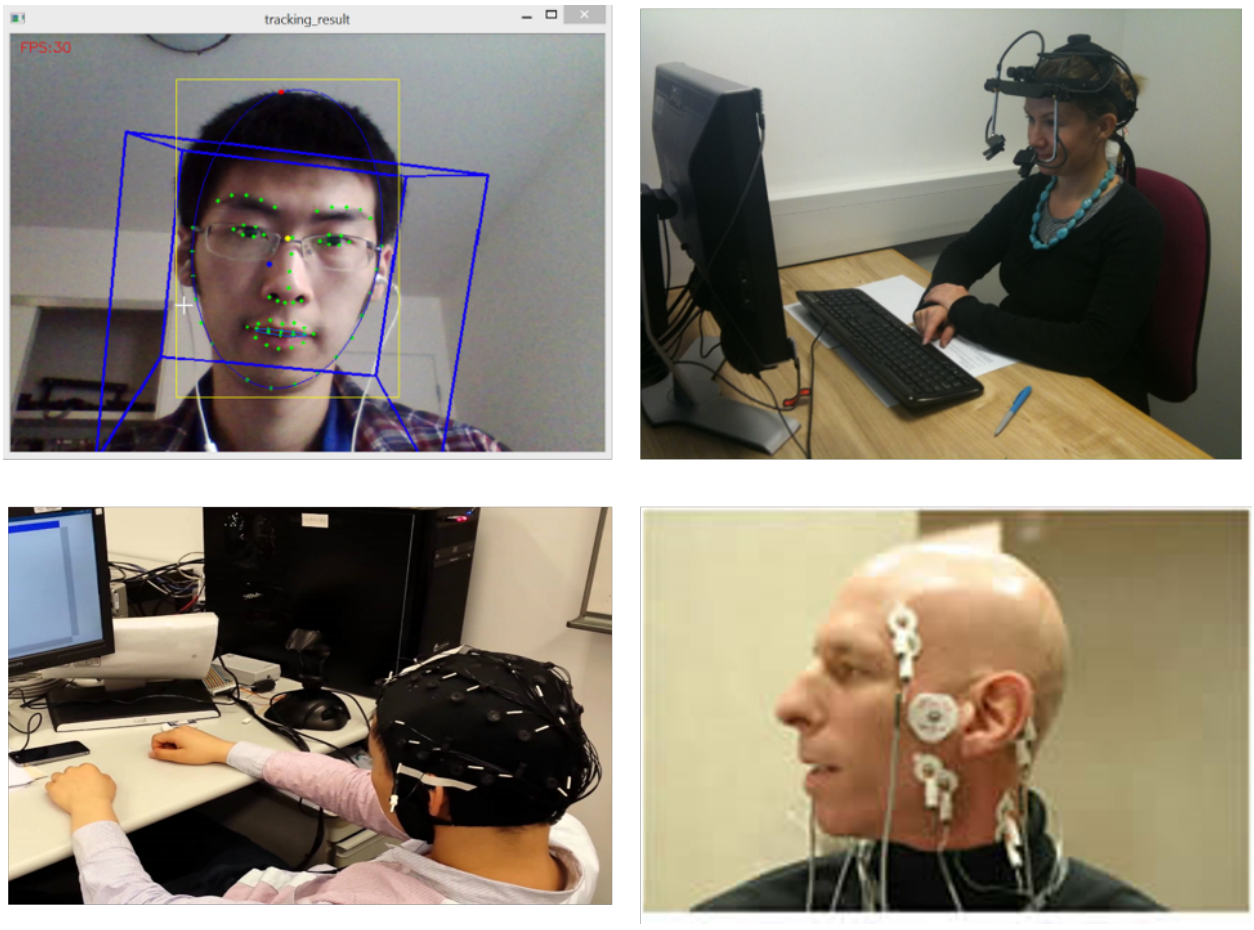

Figure 2.1: Examples of methods to control cursor position.[10, 12, 13, 3]

Chapter 3

System Design

This chapter describes our system design of the neckmouse. We first discuss our technical approach with rationale, followed by a description of the two modes of operation for controlling the mouse cursor by head tilt.

3.1 Technical Approach

We envisioned a low-cost system that is comfortable to wear, intuitive to use, and requires little or no training or calibration. To achieve these goals, we proposed a simple mechanism that requires minimal processing based on a pair of infra-red LED (IR LED) and photo detector.

3.1.1 Reflective IR Sensors

Our way of control the mouse cursor by tracking head tilt requires precise short-range distance measurement. We choose the sensing modality of active, reflective IR sensors to measure distances from each collar bone to the sides of the chin. The IR LED and photo diodes are among the

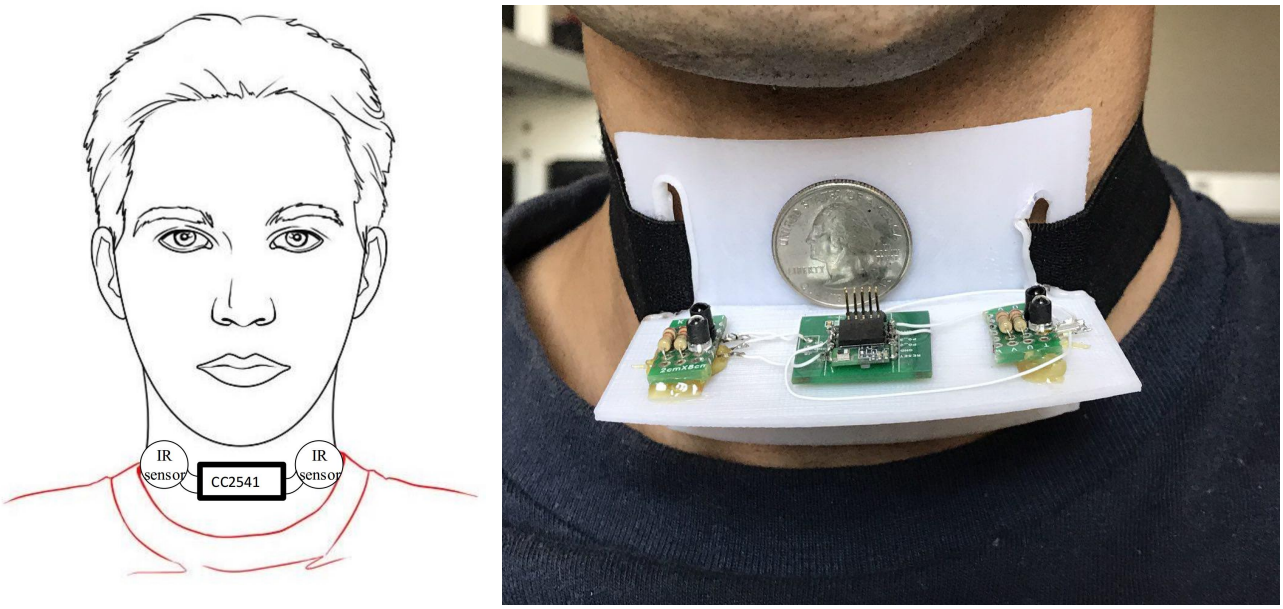

Figure 3.1: Location of the optical sensors and the CC2541 module including the BLE, ADC, and the battery. A US quarter coin is placed for size comparison but is not part of the system

lowest-cost sensors, and the rest of the functionality including the ADC, processor, and wireless communication can all be done by a commodity Bluetooth Low Energy (BLE) microcontroller unit (MCU) on the order of US$2-3 in low-volume quantities. Due to the simplicity of algorithms and calculations, they can be implemented on the same MCU to eliminate the need for third party software on the host system. As a result, the device can be directly connected to the host system using generic HID drivers.

We tested the concept on a number of users operating a prototype. Fig. 3.1 shows an overall setup of the system. The proposed algorithm for controlling the cursor on the screen is based on the distance between the IR sensor and the user's head (chin). In resting position, two sensors approximately receive the same amount of IR signal due to symmetry. Tilting down the user's head brings both sensors closer to the surface of reflection, resulting in a stronger signal. Likewise, tilting up the head weakens the reflection received by both sensors due to the longer distance. When the head turns to the right, the left sensor sees a weak reflection from the left side of the chin, but the right sensor sees little change in the reflection since the chin is still above the receiver. Thus, a weak signal on the left sensor and a strong signal on the right sensor indicates a cursor

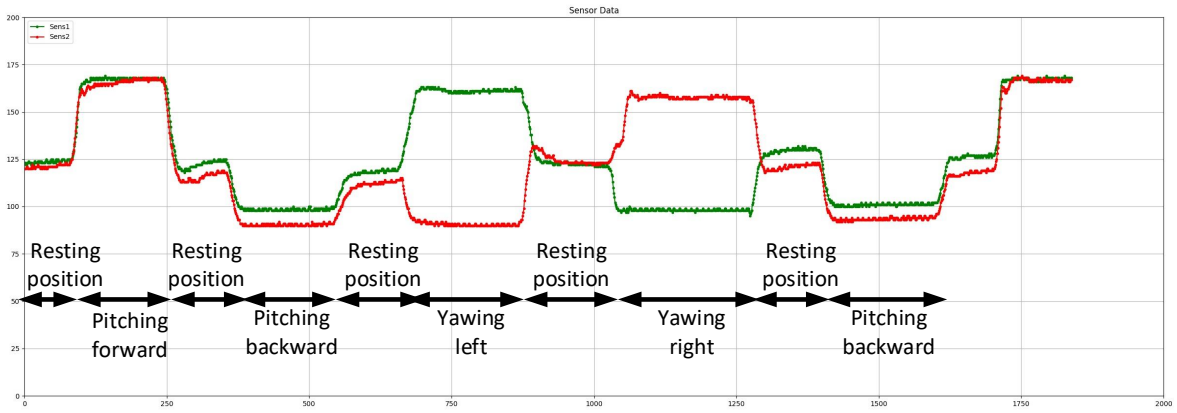

Figure 3.2: Signal changes during the pitch/ yaw movement.

movement to right. Same logic is applied for yawing to the left. Fig. 3.2 shows the signal values for different movements of pitch (up and down) and yaw (left and right).

3.1.2 Signal Conditioning

A filtering mechanism is required to eliminate environmental noise before the input signals can be used for decision making. We choose the moving average window filter in which the input data to the algorithm is mapped to the average of the last 15 data samples from the sensors. Averaging eliminates the ambient noise and helps the cursor move smoothly on the screen, preventing it from unwanted jumps. Also, due to the nature of IR radiation, input noise level can vastly change. For example, the IR level during the day or under direct sunlight is significantly higher than indoor spaces. To address this issue, we used dynamic thresholds to eliminate the problem of noise level changes. As a result, thresholds change as the noise level changes, and we will always have a steady input to the algorithm part.

3.2 Modes of Operation

The filtered signals go through a series of simple processes to map the head movement to cursor movement. The system can control the cursor in two different modes: joystick and direct mapping modes.

3.2.1 Joystick Mode

In joystick mode, the cursor moves only horizontally and vertically with a predefined constant speed in the direction of the head tilt. To determine the direction of the movement, a lower and an upper thresholds are adopted during calibration time at the initial 0.1 second of performance where the user maintains the resting head position. These thresholds are denoted by $th^{(1)}_{\text{lower}}$, $th^{(1)}_{\text{upper}}$, $th^{(2)}_{\text{lower}}$, $th^{(2)}_{\text{upper}}$, where the superscripts 1 and 2 represent the values for sensors 1 and 2. Algorithm 1 shows the pseudocode for adjusting the upper and lower thresholds for both the sensors. The averages of the raw data from the sensors in calibration mode are denoted by $S1_{\text{init}}$ and $S2_{\text{init}}$. This calibration helps us adjust the thresholds based on the ambient light and IR noise in different situations. Once the thresholds are learned, the cursor moves according to Algorithm 2.

Algorithm 1 Joystick calibration: Adjusting the thresholds.

• **Collect the samples during the calibration phase. Denote the average of the samples by** $S1_{\text{init}}$ **and** $S2_{\text{init}}$

• **Set**
$$th^{(1)}_{\text{lower}} \leftarrow S1_{\text{init}} - 5$$
$$th^{(2)}_{\text{lower}} \leftarrow S2_{\text{init}} - 5$$
$$th^{(1)}_{\text{upper}} \leftarrow S1_{\text{init}} + 10$$
$$th^{(2)}_{\text{upper}} \leftarrow S2_{\text{init}} + 10$$

Algorithm 2 Joystick mode.

- **If** $\left(S1(t) > th^{(1)}_{\text{upper}} \quad \text{and} \quad S2(t) > th^{(2)}_{\text{upper}}\right)$
  $Y \leftarrow Y - 1$; Y is the vertical cursor coordinate. Move the cursor one pixel downward.
- **If** $\left(S1(t) < th^{(1)}_{\text{lower}} \quad \text{and} \quad S2(t) < th^{(2)}_{\text{lower}}\right)$
  $Y \leftarrow Y + 1$; Move the cursor one pixel downward.
- **If** $\left(S1(t) > th^{(1)}_{\text{upper}} \quad \text{and} \quad S2(t) < th^{(2)}_{\text{lower}}\right)$
  $X \leftarrow X - 1$; X is the horizontal cursor coordinate. Move the cursor one pixel to the left.
- **If** $\left(S1(t) < th^{(1)}_{\text{lower}} \quad \text{and} \quad S2(t) > th^{(2)}_{\text{upper}}\right)$
  $X \leftarrow X + 1$; Move the cursor one pixel to the right.

3.2.2 Direct-mapping Mode

In direct-mapping mode, the absolute value of the sensor data is mapped directly to the cursor location on the GUI. If the user tilts the head up to the maximum extend, then the cursor moves to the upper border. Similarly, the extreme positions of the user's head in tilting down and yawing to the left and right are mapped to the respective borders of the GUI screen. Thus, we need to define the range of head movement so that we can map it to the pointer location. A set of training movements should be conducted in the calibration phase during the first 4 seconds of the performance. The user is required to maximally move his or her head to up, down, left, and right at startup. During this time, sensors data is being constantly checked to update the thresholds. In this mode, four thresholds are adopted for each sensor: $Th^{(1)}_U$, $Th^{(1)}_D$, $Th^{(1)}_L$, and $Th^{(1)}_R$ are specifically adopted for sensor1 in the up, down, left, and right directions, respectively. Likewise $Th^{(2)}_U$, $Th^{(2)}_D$, $Th^{(2)}_L$, and $Th^{(2)}_R$ are learned for sensor 2. Algorithm 3 shows the pseudocode for learning the thresholds in the calibration phase for direct mapping.

After 4 seconds, the thresholds are fixed and put in to the equations. There are three equations: one governs the changes in Y coordinate of the cursor, and the other two determine the X coordinate. For Y we have:

$$\text{Y} = Yaxis \times \frac{\text{avg}(S1(t), S2(t))}{\left(\text{avg}(Th^{(1)}_U, Th^{(2)}_U) - \text{avg}(Th^{(1)}_D, Th^{(2)}_D)\right)} \tag{3.1}$$

Algorithm 3 Direct mapping mode, calibration phase.

- **While in Calibration do:**
- **If** $\left(S1(t) < Th_U^{(1)} \quad \text{and} \quad S2(t) < Th_U^{(2)}\right)$
  Update $Th_U^{(1)} \leftarrow S1(t)$
  $Th_U^{(2)} \leftarrow S2(t)$
- **If** $\left(S1(t) > Th_D^{(1)} \quad \text{and} \quad S2(t) > Th_D^{(2)}\right)$
  Update $Th_D^{(1)} \leftarrow S1(t)$
  $Th_D^{(2)} \leftarrow S2(t)$
- **If** $\left(S1(t) > Th_L^{(1)} \quad \text{and} \quad S2(t) < Th_L^{(2)}\right)$
  Update $Th_L^{(1)} \leftarrow S1(t)$
  $Th_L^{(2)} \leftarrow S2(t)$
- **If** $\left(S1(t) < Th_R^{(1)} \quad \text{and} \quad S2(t) > Th_R^{(2)}\right)$
  Update $Th_R^{(1)} \leftarrow S1(t)$
  $Th_R^{(2)} \leftarrow S2(t)$

where *Yaxis* is the number of pixels in Y direction, and avg() finds the average of its inputs. For X coordinate, we had to use two equations: one controls the position in the left half of the screen and the other in the right half.

$$X = \left(1 - \frac{(S1(t) - S2(t))}{\left(Th_L^{(1)} - Th_L^{(2)}\right)}\right) \times \frac{Xaxis}{2} \tag{3.2}$$

$$\mathrm{X} = \frac{(S2 - S1)}{\left(Th_R^{(2)} - Th_R^{(1)}\right)} \times \left(\frac{Xaxis}{2}\right) + \frac{Xaxis}{2} \tag{3.3}$$

As an example, if the user turns the head left to the maximum extend, we have $S1 = Th_L^{(1)}$ and $S2 = Th_L^{(2)}$, resulting in $X = 0$. If the head stays in the middle, we have $S1 = S2$, resulting $X = \frac{Xaxis}{2}$. The same concept applies to the last equation.

3.3 Click function

The aforementioned method proved to be working fine for cursor movement, but a full-function mouse device must support clicking as well, not just pointing. We had the options of building a separate sensing unit just to handle clicking or of integrating clicking into the main cursor tracker. At first, we decided to make a second device that could be mounted on the finger to detect certain patterns and translate it to left- and right-click commands. Such a device would need a complete processor unit, sensors to detect movements, and a communication module to interact with main device, and it could double the hardware requirement. We considered the alternative solutions of detecting clicking pattern by adding extra sensors that can be processed by the same MCU with marginal overhead.

3.3.1 Capacitive Touch Sensor

The first option considered is a small capacitor-based touch sensor on top of the main board on the deck that could be trigged by slight touch of the skin. To enable clicking, the user is required to tilt the head down, but unfortunately, this gesture can be easily confused with the move-down command for the cursor. In fact, almost every move the head to operate the capacitive touch sensor could interfere with the positioning algorithm.

3.3.2 Force-Sensing Resistors

Our chosen solution is to use two FSR sensor, each positioned on each of the two cheecks. An FSR (force-sensing resistor, also known as force-sensitive resistor) is a resistor whose resistance depends on the shape and force applied to the sensor. To issue a click, the user can warp the FSR simply by either inflating the cheek or poking the tongue at it. FSR sensors can be set up just like infrared receivers, using a serial resistor with proper value. The whole system (FSR and resistor)

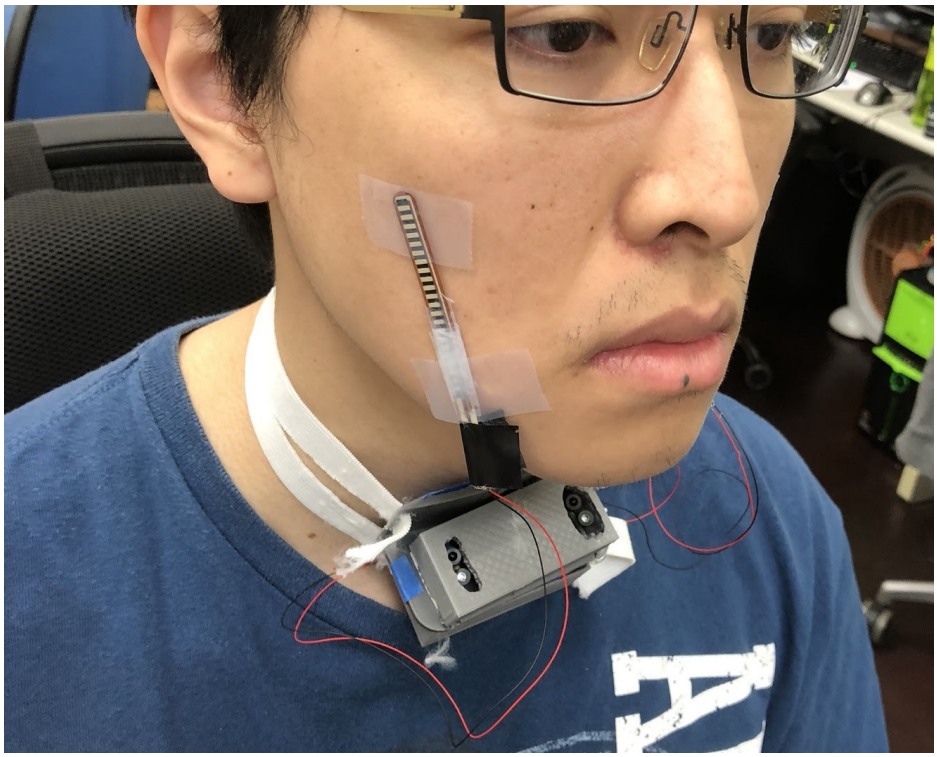

Figure 3.3: FSR sensors on the chick to detect clicking.

can produce variable output voltage that can be captured by the built-in ADC of the MCU. By mounting FSRs on the two cheeks, we can implement both the left and right clicking function.

Chapter 4

Implementation

This chapter describes an implementation of the proposead neckmouse. Fig. 4.5 shows the block diagram of the proposed system. The wearable system is centered around an MCU with an on-chip wireless communication interface, the two optical sensor-transmitter pairs, the force-sensing resistors, and the power source. These subsystems are explained in the following sections.

4.1 MCU and Wireless Communication

Our implementation is centered on an MCU and wireless communication. On one side, the MCU reads signals from the sensors for detecting the head position. The MCU is programmable and can process data. It can send either raw data or processed mouse commands to the host system via its wireless communication module. Many conventional MCUs can be used. To be suitable for a wearable device, we choose an MCU with more integrated features, including on-chip analog-to-digital converter (ADC) for interfacing with the optical sensors, and low power consumption, rather than one with high performance. The choice of wireless communication enables ease of use and mobility of device. The device and the host need to support a compatible protocol. We

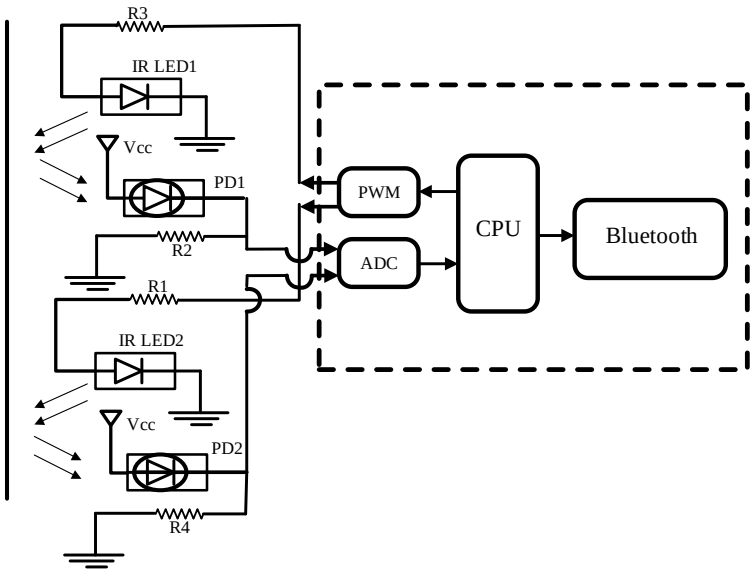

Figure 4.1: The system block diagram.

choose Bluetooth Low Energy (BLE) Technology, suitable for low-data-rate communication. It consumes considerably less power compared to conventional Bluetooth modules. Moreover, BLE is integrated on a number of popular MCUs, making it low cost and compact.

Our choice of such RF-MCU is the TI CC2541 for its maturity and availability of prototyping boards in our lab. The processor is an 8051-compatible core surrounded by 8-KB SRAM, 256 KB flash, 8-channel ADC, timers, UARTs, I2C, and SPI on the chip. It also includes a BLE transceiver that enables connection to any Bluetooth device including the user's computer.

4.2 Optical Sensor

Each of our two optical sensors is implemented using a pair of IR LED and photodiodes for estimating the distance of the LED from the edges of user's chin. This section describes the IR transmitter and receiver pair, followed by the circuit for measurement.

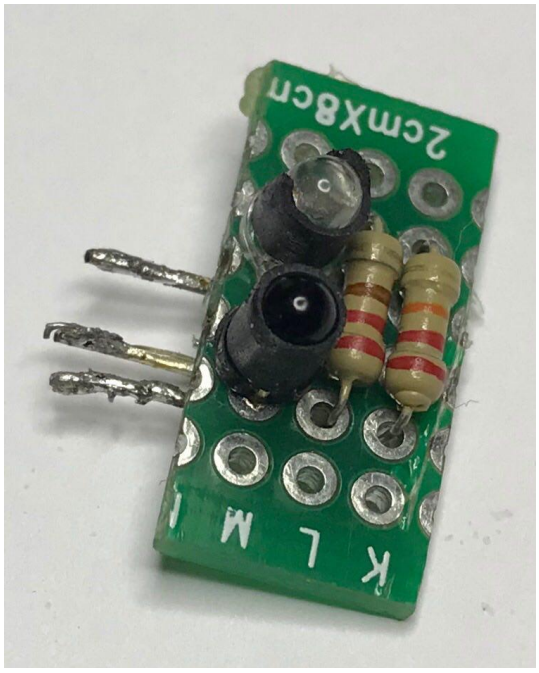

Figure 4.2: Actual circuit of the sensor board.

4.2.1 IR Transmitter-Receiver Pairs

The IR transmitters and receivers work in the 940 nm wavelength and are available commercially off-the-shelf. For simplicity, the transmitter is set to be voltage-driven rather than current-driven using a 220 Ω shunt resistor to control the current and IR brightness as shown in Fig. 4.5.

The receiver is an IR photo-detector, modeled as a variable resistor. The resistance is determined by the amount of IR that the photo detector receives. A shunt resistor of 22 KΩ is used to bias the output. The output voltage changes approximately linearly with respect to the intensity of the IR light. The IR intensity itself is a function of the distance between the sensor board and the reflective surface (chin). Each sensor can detect the chin distance in the range of 0.2 to 4 inches (0.45 cm to 9 cm).

Code 4.1: Device code

```
void SensorRead()
{
  int16 data[4];
  int16 adc;
  //configure ADC2 and ADC3 pins:
  APCFG |= 0x0C;
  //HAL_ADC_REF_VOLT:
  HalAdcSetReference (ADCCON3_EREF_AVDD);

  enable_sensors();
  delayms(1);

  adc = HalAdcRead( HAL_ADC_CHN_AIN3, HAL_ADC_RESOLUTION_10 );
  data[0] = ((adc>>1) & 0xFF);

  delayms(1);
  //Read sensor2:
  adc = HalAdcRead( HAL_ADC_CHN_AIN2, HAL_ADC_RESOLUTION_10 );
  data[1] = (adc>>1) & 0xFF;
  disable_sensors();
}
```

Figure 4.3: Source code for reading data from sensors.

4.2.2 PWM Control and ADC

To make the device power efficient, pulse-width modulation (PWM) is used. Sensors are powered on once every 50 ms for the duration of 1 ms; in other words, 20 Hz sampling frequency and 5% duty cycle. This reduces the average power consumption considerably and increases battery endurance without significantly affecting the device's performance.

Since the output signal is analog, it can have any value in the range of VCC (3.8 V) and GND (0 V). The signal is converted using the on-chip ADC on the MCU. The ADC samples each photo diode once every 50 ms at 8-bit resolution using a time-multiplexing scheme. It means different analog channels are selected one at a time to be converted by a single ADC.

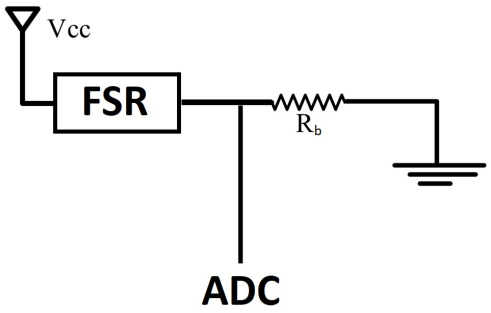

Figure 4.4: Sensor circuit to detect left/right click.

4.3 FSR buttons for clicking

To detect click command, we chose to mount a Force Sensitive Resistor (FSR) on each of the two cheeks. The sensors can be trigged by inflating cheeks or simply poking them with the tongue. Based on how much it is bent, it can have a variable resistance in the range of 25KΩ to 42.5KΩ. By serializing it with a 49KΩ bias resistor, it has output signal in the range of $V_O = V_{\text{cc}} \times \frac{25k}{25k+49k} \approx \frac{V_{\text{cc}}}{3}$ to $V_O = V_{\text{cc}} \times \frac{42.5k}{42.5k+49k} \approx \frac{V_{\text{cc}}}{2}$. This is a considerable change that can be captured clearly by an 8-bit ADC. Fig. 4.4 shows the diagram for click detection sensor.

4.4 Power Subsystem

The power subsystem consists of a battery and a power regulator. We chose a rechargeable lithium-polymer battery as the source of power. It is fed to a buck converter, the TPS62740DSS, to reduce the voltage from 3.8 V to 2.5 V to be used by the MCU. Note that IR transmitters are directly connected to 3.8 V from the battery, rather than the regulated 2.5 V, as the higher voltage results in more accurate distance measurement. To implement the PWM function, the cathode pin of the sensors are connected to the MCU's general-purpose input/output (GPIO) pins. When these pins are on the zero part of the PWM, sensor boards will be enabled. Since the most of power is consumed by the BLE and IR LEDs, applying PWM on the IR LEDs results in reduction of the

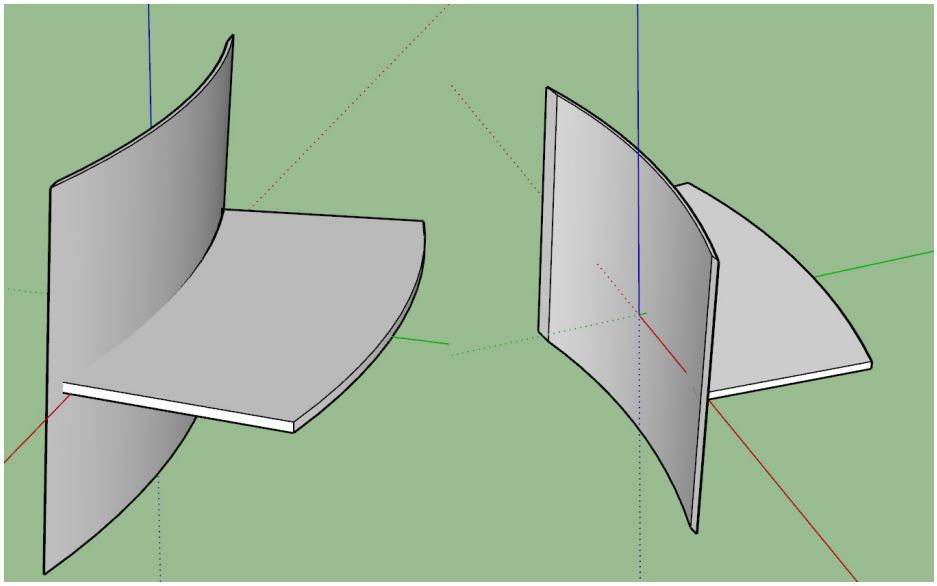

Figure 4.5: Main deck design.

average power consumption down to 10% in the sensor section, i.e., from 16 mA to 1.6 mA for the two sensor boards. The battery we are using for the system has a capacity of 500 mAh and is mounted under the main deck. It can be recharged by wire using off-the-shelf chargers. Wireless charging can be considered for future work.

4.5 Mechanical Design

All of the components including the sensors, main board, and the battery are placed on a curved-shaped deck that can easily be mounted on the neck as in Fig. 3.1. We designed and 3D-printed this deck with overall size of $4 \times 7 \times 3$ cm^3. To make the device compatible with different neck sizes, two fabric straps are mounted on either side of the deck that can tighten the device to different neck sizes. To reduce environmental noises, a plastic cover is mounted on the IR transmitter and receiver. This helps the receiver block out most sources of IR noise while detecting the IR reflection from the chin.

4.6 Data Processing

A proof-of-concept version of the system transmits raw data to the host computer, which runs a program to process and display the cursor. A more mature implementation works just like a plug-and-play Bluetooth mouse to a PC by processing the sensor data locally and transmitting mouse events in the human interface device (HID) profile for Bluetooth.

4.6.1 On-Host Processing

In the initial version, raw data is sent directly to the host system, where all the processing and algorithms are implemented. This enables easier exploration and better reproducibility, as we can run a variety of algorithms on the same recorded data. Depending on the mode of operation, either the joystick or direct-mapping algorithm is invoked to process the data as mouse commands. We have developed a Python-based Graphical User Interface (GUI) for the PC to receive data and render the corresponding position on the screen.

4.6.2 On-Node Processing

We have also implemented the human interface device (HID) profile over BLE. The HID profile enables our device to work as a wireless mouse over BLE and enables plug-and-play operation without requiring driver installation. It processes the data on the device and send mouse movement data to the host just as a mouse would in the form of an HID report in BLE. Most of this code can be written based on sample code provided by the vendor of the MCU [11]

Code 4.2: Device code

```
#---setup to turn on peripheral notification-----
print("setup notify...",conn.getServices())
svc = conn.getServiceByUUID("0000fff6-0000-1000-80...
  ...00-00805f9b34fb")
ch = svc.getCharacteristics("0000fff7-0000-1000-80...
  ...00-00805f9b34fb")[0]
conn.writeCharacteristic(ch.handle + 2,...
  ...struct.pack('<bb', 0x01, 0x00), True)
t = time.time()      #start timer
print("start time:",t)
make_target("random")
while True:
  try:
    if conn.waitForNotifications(0.001):
      #get raw data from BLE message
      rawdata = [conn.sensor[0:4],conn.sensor[4:8]]
      #extract raw data value
      data = Uint2Toshort(rawdata)
      sys.stdout.flush()
      if (first_time_flag != 1):
      #calibration on the start up
        calibration(data[1],data[0])
      else:
      #calculate position based on mode of operation
        modeSelect(mode,data[1],data[0])
      #check if cursor is in the target area
        check_target()
  except:
    print("fail notify")
    break
```

Figure 4.6: Source code for reading data from sensors.

Chapter 5

Results and Discussion

We experiment with our headmouse implementation by conducting a series of tests and evaluate its performance according to a number of metrics. We compare the results and assess the efficiency and user friendliness of our proposed system.

5.1 Experimental Setup

Two different users volunteered to test the system in two modes. The task requires the user to move the cursor into a target. The location of the target on the screen and its size are randomly chosen, and the task is repeated 50 times for joystick modes and for direct-mapping mode. We vary the *index of difficulty*, which is a function of the location and the size of the target. The GUI automatically logs the *move time* (*MT*) and the *path length P*. The user tries to follow a pre-drawn pattern on the screen to reach the target as depicted in Fig. 5.1. The path traveled by the cursor as controlled by the user is rendered in a black line. In Fig. 5.1(a), the user is following the path in red, while in Fig. 5.1(b), the user tries to move the cursor inside the target depicted with a square. For the test to be considered successful, the cursor should stay inside the target region for two

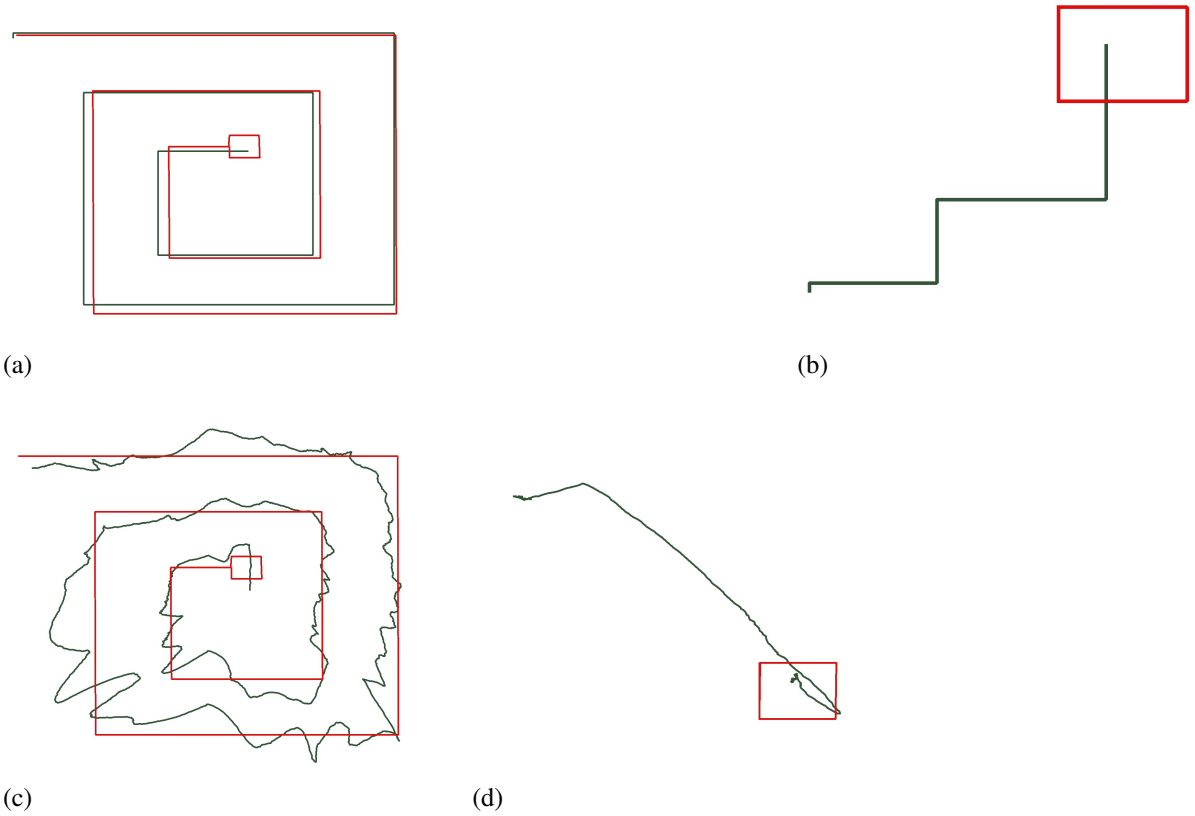

Figure 5.1: The testing platforms, (a) User tries to follow a predefined path in the joystick mode. (b) User tries to guide the cursor into a target in the joystick mode. (c) Following the path in direct mapping mode. (d) Guiding the cursor to the target in direct mapping mode.

seconds. As explained before, joystick mode requires no effort to calibrate, but in Direct mapping mode, the user is required to move the head around in the first 5 seconds of operation to feed the device with movement boundaries.

5.2 Performance Metrics

Fitts's law is a predictive model of human movement primarily used in human-computer interaction and ergonomics. This scientific law predicts that the time required to rapidly move to a target area is a function of the ratio between the distance to the target and the width of the target. Fitts's

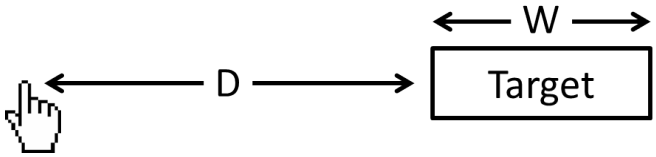

Figure 5.2: Demonstrating Fitts's law.

law is used to model the act of pointing, either by physically touching an object with a hand or finger, or virtually, by pointing to an object on a computer monitor using a pointing device [4]. According to Fitt's law, there is an inherent trade-off between the speed and accuracy. We resort to the performance metrics mostly used in the state-of-art literature to quantify the performance. These metrics, including the index of difficulty, path efficiency, throughput, and overshoot are defined as follows:

Index of Difficulty (ID): ID is defined as

$$ID = \frac{D}{W}, \tag{5.1}$$

where D is the distance between the original location of the cursor and the center of the target, and W is the width of the target. A higher index of difficulty means more control over the pointing device is required to navigate the cursor into the target.

Path Efficiency (PE): Path efficiency is a measure of straightness of the path the cursor is traveling and is defined as the ratio of the total distance between the original location of the cursor and target to the path length traveled by the cursor. That is,

$$PE = \frac{D}{P}, \tag{5.2}$$

where P is the length of the path traveled by the cursor.

Throughput (TP): Throughput is a metric for measuring how fast the cursor is moving and is

defined as:

$$TP = \frac{ID}{MT}, \tag{5.3}$$

where MT is the time it takes for the user to move the cursor to the target zone.

All these metrics are calculated automatically for each test case and the results are discussed in the following section.

5.3 Results

After collecting data from the experiments, we develop a MATLAB function to calculate and plot the results. Fig. 5.5 shows the path efficiency of the users for both joystick and direct-mapping modes, while Fig. 5.3 plots the throughput. Although the results are not uniform, clear patterns can be seen in the index of difficulty over all data set. These results confirm that headmouse can be used as a substitute for conventional mouse systems. Although relying solely on head to guide the pointer can cause fatigue in neck muscles after long use, increasing the device sensitivity can eliminate the problem to some extend.

As can be observed, path efficiency is higher in joystick mode for both users. Although one would expect path efficiency to to decrease as the index of difficulty increases (since ID increases by increasing the distance between the cursor's initial location and the target or by decreasing the size of the target), Fig. 5.5 shows otherwise; at some points, path efficiency actually increases with index of difficulty. This is because the device provides users with fine-tuned control over the cursor position. As result, distance alone is not effective on path efficiency.

In fact, the angle between the X or Y axis and the line from starting point and end point matters more than the distance. For instance, if the required move to reach to destination is only on X axis

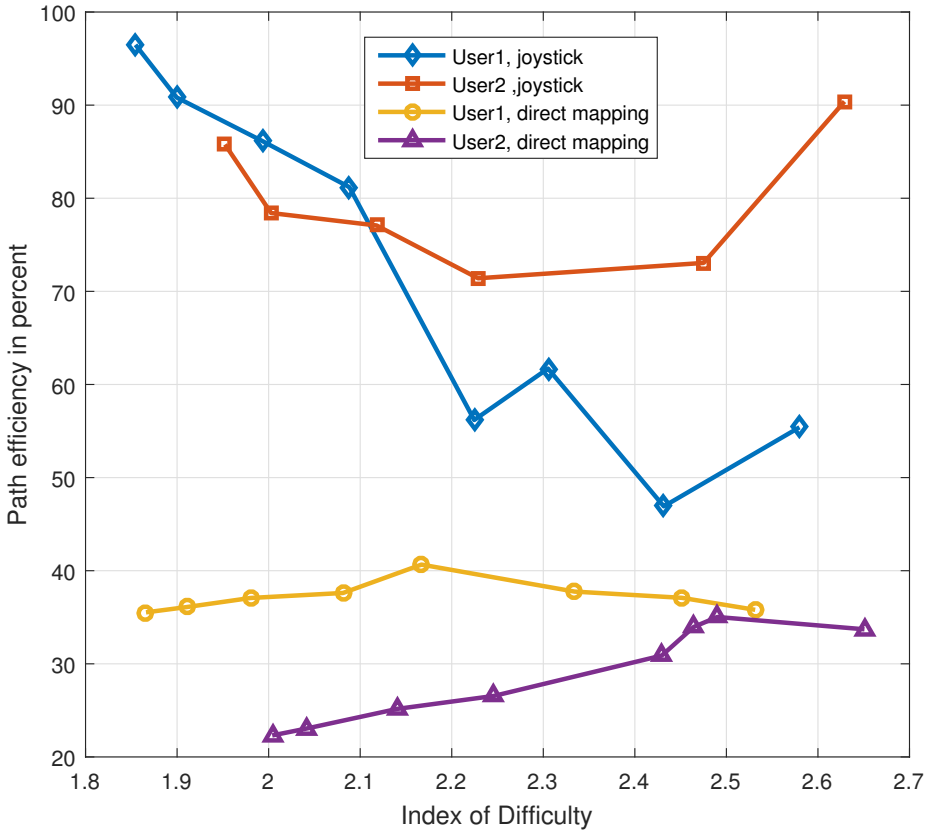

Figure 5.3: Path efficiency with respect to index of difficulty for both modes.

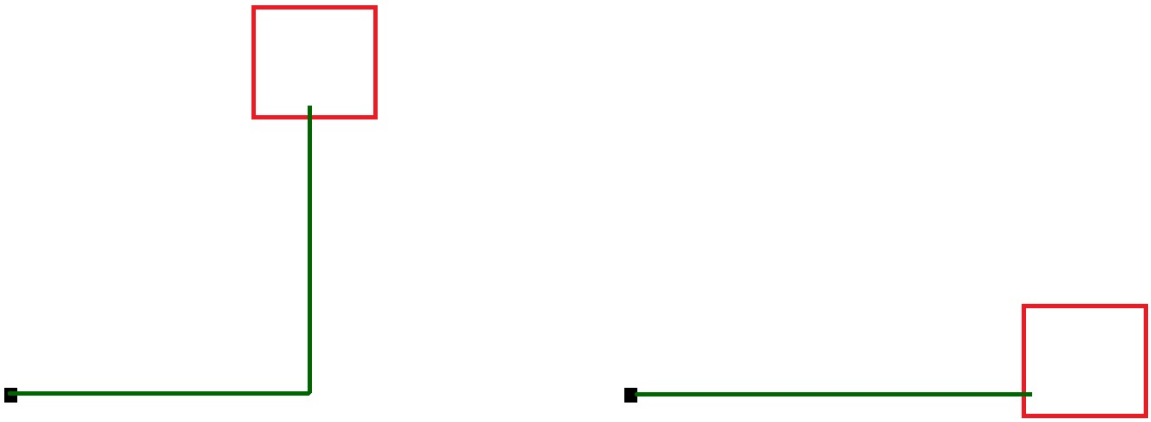

Figure 5.4: Example of same measures with different index of difficulty.

or Y axis, we will have $D = P$ where D is the actual distance and P is the traveled path; but if the angle is at 45 degree, with distance of D, the cursor have to travel for $D\sqrt{2}$.

Fig. 5.4 shows two different examples with thesame index of difficulty (i.e., same distance of 10 cm and target size of 2.7 cm). The traveled paths for Fig. 5.4(a) and Fig. 5.4(b) are 10 and 14, respectively. In direct mapping mode, the size also matters since in case of small target, users can overshoot, which means the cursor passes the target and has to travel back. This increases P and decreases the path efficiency.

According to Fig. 5.5, again, joystick mode has better throughput than direct-mapping mode does, and it increases with index of difficulty. It is also observed that the more users try the experiment, the better result they get. One explanation is that they are getting used to the system and its responsiveness. In both figures, user 2 performed better than user 1 did, most probably because user 2 had tried device before the test for a few times.

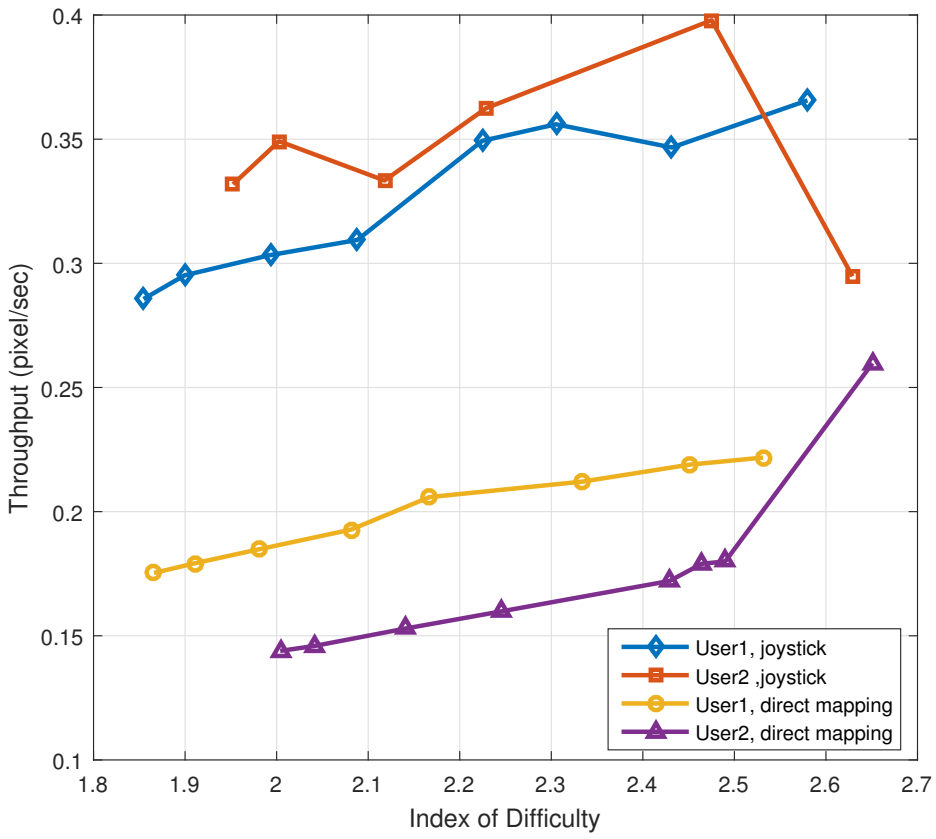

Figure 5.5: Throughput with respect to index of difficulty for both modes.

Chapter 6

Conclusions

This thesis has proposed a head-controlled assistive wireless mouse for users suffering from motor disabilities in their hands and fingers. The control of the mouse cursor position is done entirely by head tilting, and clicking is done by either inflating the cheek or poking the tongue on the cheek. The system uses a pair of reflective IR to detect relative collar-to-undercheek distances for 2D cursor positioning and a pair of force-sensing resistors (FSR) for clicking.

The advantages of our system include cost, ease of operation, and comfort. The very simple yet robust sensing mechanisms and wireless-enabled MCU enable the system to be constructed at very low cost compared to alternative assistive technologies such as voice command, eye tracker, head tracker, tongue tracker, EMG, and EEG based trackers. The use of Bluetooth Low Energy (BLE) as an industry-standard plug-and-play protocol makes it compatible with virtually all modern personal computers without additional software or driver installation. The system is relatively intuitive to operate and can be controlled with good precision. The opitcal sensors are non-contact on the collar and thus minimally intrusive.

A number of limitations with our system include the potential intrusiveness of the FSRs and interference from ambient light. The FSRs are attached externally on the cheecks, and while they are

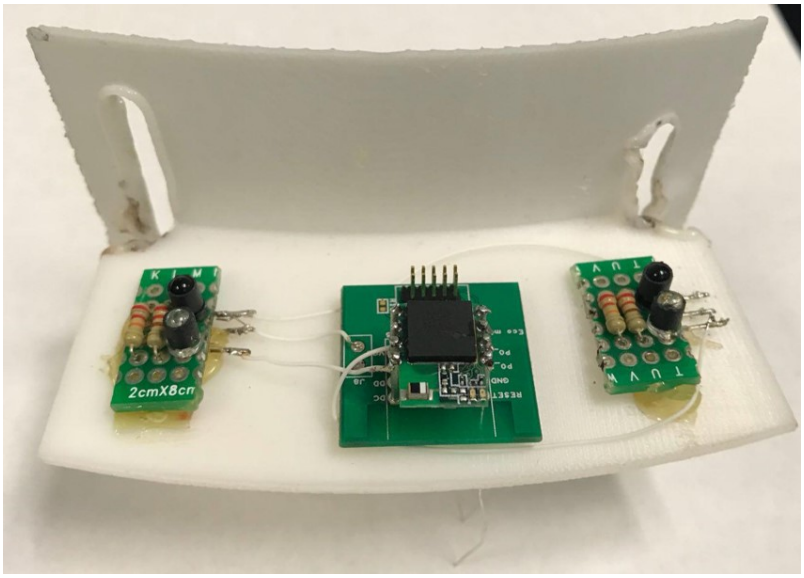

Figure 6.1: Headmouse device.

intuitive and effective to operate, they have intrusive appearance. However, we believe that these appearance issues can be solved by industrial design, which we leave as future work.

Another direction for further work is the mode of operation. We currently support both joystick and direct-mapping modes. Although joystick mode gives more precise control in our test cases, it is less efficient to operate than direct-ampping mode. Direct mapping, on the other hand, requires calibration, which we would like to eliminate entirely. Better signal processing and fine tuning will make direct-mapping mode more usable and intuitive.

Appendix A

Schematic of Ecomini

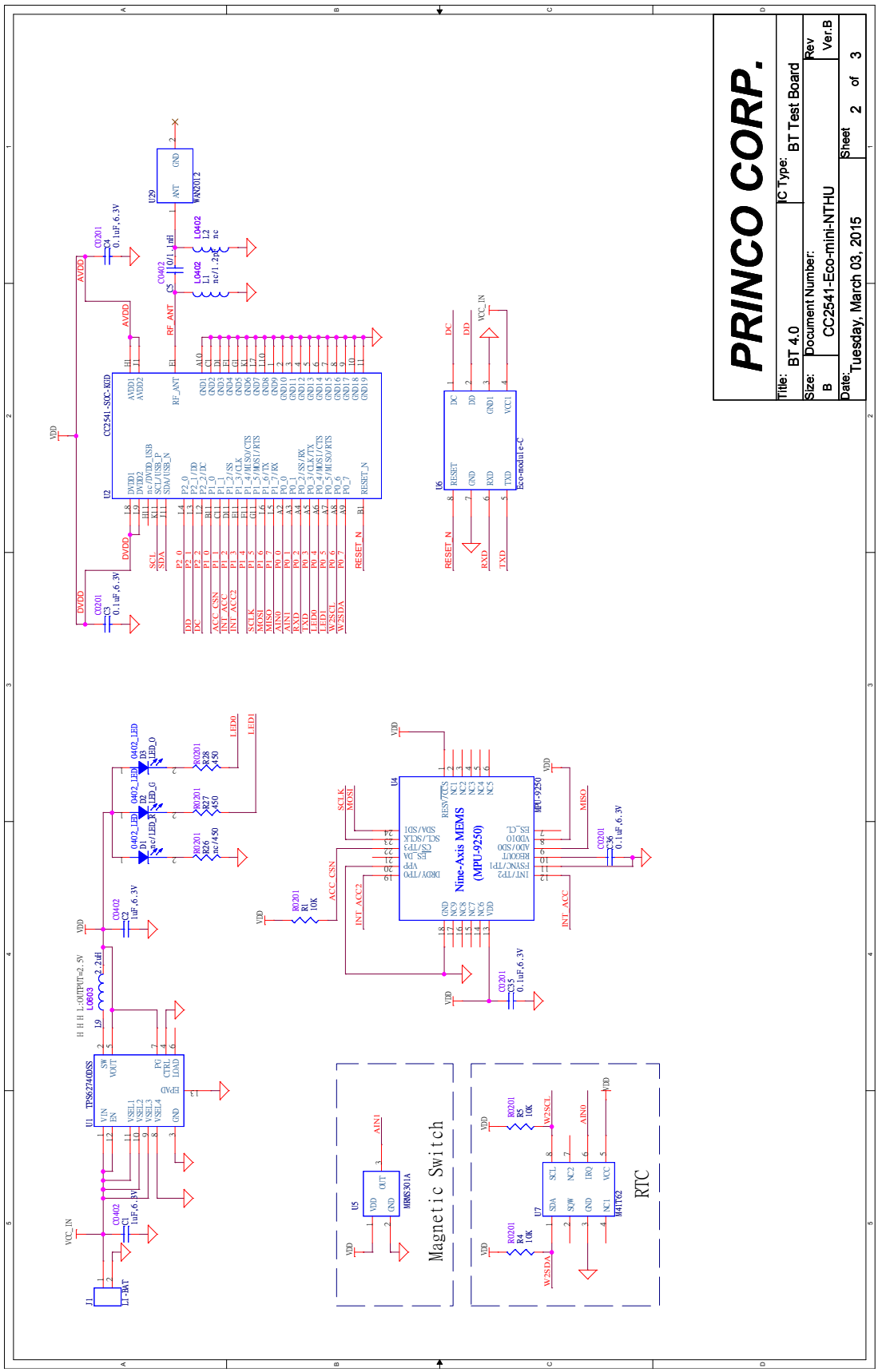

Figure A.1: Ecomini, uses as the main part of the system to collect signals and send it through BLE.

Bibliography

[1] A. Al-Rahayfeh and M. Faezipour. Eye tracking and head movement detection: A state-of-art survey. *IEEE Journal of Translational Engineering in Health and Medicine*, 1:2100212–2100212, 2013.

[2] Amputee Coalition. Limb loss statistics. https://www.amputee-coalition.org/limb-loss-resource-center/resources-filtered/resources-by-topic/limb-loss-statistics/limb-loss-statistics/.

[3] C. A. Chan. Diagnostic technology. `http://www.claytonchandds.com/diagnostictechnology.html`.

[4] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 121(3):262–269, October 1992.

[5] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:478–500, Mar. 2010.

[6] X. Huo, J. Wang, and M. Ghovanloo. A magneto-inductive sensor based wireless tongue-computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(5):497–504, Oct. 2008.

[7] Industrial Safety & Hygene News. Statistics on hand and arm loss. `https://www.ishn.com/articles/97844-statistics-on-hand-and-arm-loss`.

[8] C. McCrimmon, J. Fu, M. Wang, L. Silva Lopes, P. Wang;, A. Karimi-Bidhendi, C. Liu, P. Heydari, Z. Nenadic, and A. Do. Performance assessment of a custom, portable and low-cost brain-computer interface platform. *IEEE Transactions on Biomedical Engineeringr*, PP(99), 2017.

[9] D. J. McFarland, D. J. Krusienski, W. A. Sarnackia, and J. R. Wolpaw. Emulation of computer mouse control with a noninvasive brain-computer interface. *Journal of Neural Engineering*, 5(2):101, 2008.

[10] Y. Qin and Y. Wang. Virtual reality ball game. `http://eaglesky.github.io/VRBallGame/`.

[11] Texas Instruments. CC2541 Bluetooth Smart Remote Control Kit. `http://www.ti.com/tool/CC2541DK-RC`, 2017.

[12] University of Kent. Psychology research: Eyetracking lab. https://www.kent.ac.uk/psychology/research/facilities/eyetrack.html.

[13] H. Wang. EEG based brain machine interface–online 2D cursor control experiment. https://www.youtube.com/watch?v=NmZXruqbTVs.

[14] M. R. Williams and R. F. Kirsch. Evaluation of head orientation and neck muscle EMG signals as command inputs to a human-computer interface for individuals with high tetraplegia. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(5):485–496, Oct. 2008.