Biting Banter

Leiden University Institute of Advanced Computer Science

Media Technology MSc. 2024

Hardware & Physical Computing

Boet Bouten (s2829185), Nataliia Kaminskaia (s4027078), Trent Eriksen (s3746887)

Initial plan

The conception of this project started out with the idea of working with haptics. With this idea in mind we started looking for possible installations which we could build that centered around a sense of touch. From here we took inspiration from a study done by Hart and colleagues which revolved around building a communication tool for deaf and blind individuals based on Morse code (Hart et al., 2016). By biting on a force probe, one individual could send Morse signals that would be received by the recipient in the form of short and long bursts of vibrations corresponding to dits and dats. We decided to base our project on this principle of haptic Morse code communication.

As such, the original plan was to encapsulate a pressure sensor into a mouthpiece which a user could bite on safely. The input from the pressure sensor would then activate a vibrating actuator which would be tied to some kind of ornament, for which we decided a bracelet would be most suitable. The code in Arduino would then convert the bite-inputs into letters based on set time intervals for 'short' and 'long bites'. The point of the installation would be for the recipient (the user wearing the bracelet) to decipher, with the help of a Morse code dictionary, the message of the sender (the user biting on the mouthpiece). Originally we thought that we would need both Pure Data and Arduino code to build this installation, a consideration which we included in our original project proposal. We made a general initial set up for the project in Figma which is depicted in Figure 1.

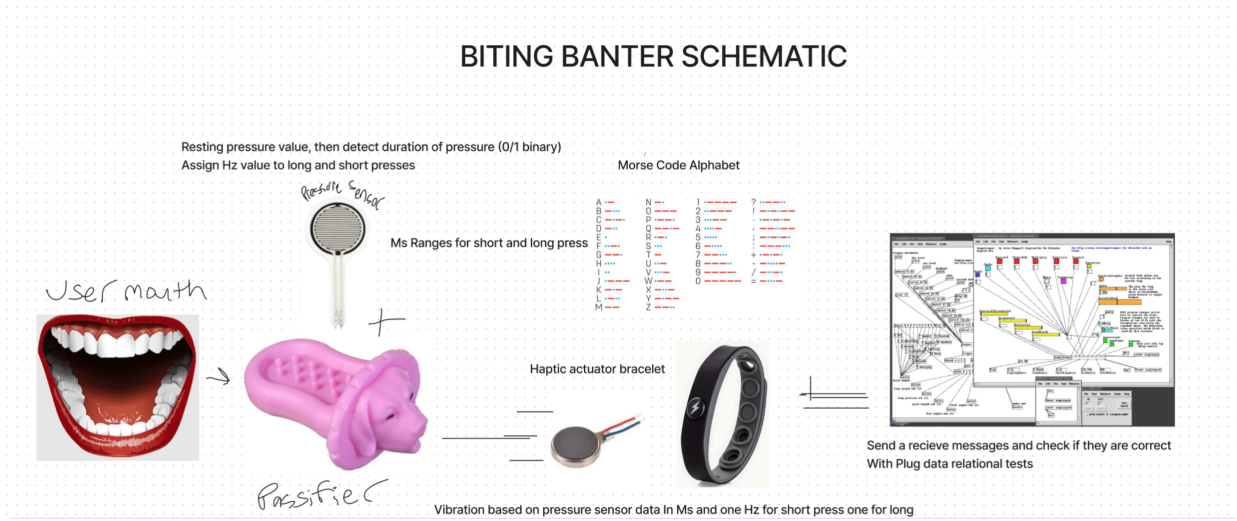


Figure 1.

The most essential parts we would need for this setup were pressure sensors, vibrating actuators, a LCD-screen to display the letters converted from the pressure sensor input, a bracelet which we could attach one or multiple actuators to, and a mouthpiece which would both be safe to bite on and would not damage the pressure sensor in any way. We would also need wires of appropriate length and implement them in such a way that there would be no electrical hazards for the users. We made lists of all the testing necessary for each part of the installation which are laid down in Figure 2. The same day we ordered all of the necessary parts and got to work with it in the first class afterwards.

Project Planning

Project Order of Operations

- Determine and Order Parts (12/12/23)
- Test Sensors / Actuators (12/31/23)
- Test Bracelet / Pacifier (12/31/23)
- Data Mapping (1/10/24)
- Data Rules (1/10/24)
- Whole System Test / debug (1/15/24)
- Presentation (1/16/24)
- Documentation (1/23/24)

Purchase Orders

- Determine Pressure Sensors > Purchase 12/12
- Determine Actuators (Piezo-resistive) > Purchase 12/12
- Determine Pacifiers (eraser / silicon mold) > Purchase 12/12
- Plastic Wrap for hygeine
- Haptic bracelet?
- LCD screen > Purchase 12/12
-

Sensor Testing (pacifier bite pressure)

- Test for / establish resting values
- Test pressure sensors for force range (skip if 0/1)
- Test pressure sensors for long/short press
- Test capturing pressure force and duration data in Arduino
- Test sending pressure force and duration data from Arduino to Pd

Actuator Testing (haptic bracelet feedback)

- Test for / establish resting values
- Test haptic actuator for Hz range (low/high intensity vibration remove if 0/1)
- Test haptic actuator for long/short press
- Test haptic actuator for receiving Hz data in Arduino from Pd
- Test translating sensor force (PSI to Hz) and duration data (both in ms) for actuator to reproduce (Pd / Arduino)

Data Mapping & Rules (Arduinio, PlugData)

- Determine normalized bite pressure range values (Arduino) (exclude if 0/1 bite/no bite) (*STATE ENGINE*)
- Determine normalized bite duration range values (Arduino)
- Set up Morse rules to indicate duration value combinations for each letter (Pd/Arduino) ie (250ms, 750ms) = Letter 'A'
- Set up text output to indicate which letters where "spelled" with "Oral Morse" (Pd/Arduino)
- how to know when to start the morse code phase and stop? do we need a button or switch to indicate active / inactive?

Figure 2.

Who did what?

From a practical point of view almost all of the work done for this project was done as a team. We did decide to split our parts amongst the three of us before the Christmas break so we could each work on separate aspects of the installation during this time, but other courses took priority over the Holidays. Generally speaking we did not make much of a division of tasks, instead opting to work through each step of the project together and make decisions in real time as to who would do what during a session. A benefit of this way of working was that all three of us were up to date about the state of each aspect of the project (coding, hardware and design) at all times which made teamwork efficient. Most of the work on this project was done at the faculty during class, although we did also meet outside of class hours at the University Library to work on it together.

We all met and brainstormed for several hours and initially agreed we wanted to work on something with reactive haptics. Nate (Nataliia) found a research (Hart et al., 2016) paper on Morse code with bite force and haptics that was intended to be used by hospitalized individuals that were going blind and deaf at the same time and who had suffered from muscular degeneration to the point of partial or full paralysis. From this we all collectively built a Figma board that described the basic outline for a low fidelity prototype based on sending signals with a pressure sensor to a haptic actuator. From establishing this signal flow we all collectively sourced the proper materials to build what we included in this project plan. This included research into the proper sensor types, actuators, oral devices, LCD screens and many other parts. Once receiving all sourced parts we all collectively built the prototype step by step which first included testing of all equipment for proper functionality. Once this was confirmed we worked on the code together, getting some assistance from teaching assistants. After a few iterations we confirmed the arduino code worked and it was time to finish building the housing for the unit. Nate and Boet found and sourced the outer housing and other decorative elements that reflected the spirit of the project while also encouraging users to engage with our project. We all together completed the construction of the prototype to ensure mechanical stability and aesthetic function to be fully prepared for public presentation.

Problems you ran into

While undertaking our project, we initially anticipated more complications. Notably, after discussions with Mikkel, we were relieved to learn that Pure Data could be entirely omitted, allowing all coding to be done in Arduino. This significantly simplified our process, as we had previously considered the integration of Pure Data a major obstacle. Additionally, the extensive history and resources available on Morse code greatly aided in its implementation within our programming framework.

Throughout the project, we encountered several minor issues. When faced with challenges beyond our expertise, we sought guidance from our teaching assistant Mikkel or our teacher Paul. A particular challenge was protecting our force-sensing resistor (FSR), intended for biting interaction, from excessive force and humidity. Our initial idea of creating a silicon mold was set aside due to its complexity. Instead, we found a practical solution in repurposing an AirTag case, which perfectly fit the FSR. This case was then modified to serve as an effective sensor cover. We also addressed hygiene concerns for participants by using small ziplock bags, similar to those used for dental scans, allowing for up to 100 participants to safely interact with the project.

The actuator's implementation was approached with subtlety in mind, leading us to adopt a bracelet design akin to an Apple Watch. Utilizing velcro cable ties, we secured the actuator to the inner wrist, achieving a discreet and sensitive vibrating bracelet.

The coding aspect presented its own complexities, particularly in translating Morse code signals to letters on the screen. Given the challenge of maintaining consistent signal durations by inexperienced users, we adjusted the Morse code values to more user-friendly thresholds. By the project's conclusion, we were able to accurately translate live Morse code communication into on-screen text among the team members.

In preparation for our presentation, we focused on constructing a more stable version of our prototype. This necessitated the use of longer wires for comfortable operation. Our search for appropriate wires led us to choose "signaalkabel," composed of four 0.5 mm^2 wires encased in plastic, which we then adapted for use with two actuators and the biting sensor. Despite these challenges, our project's successful completion showcased our adaptability and resourcefulness.

Results

The end result of our project is a haptic game based on Morse code communication that's meant to push participants out of their comfort zone and open themselves up to an unusual and thought-provoking means of communicating with each other. While inspired by research originating from the medical field, we aim for our product to offer a fun and interesting experience more so than it being a practical communicative tool (Hart et al., 2016). The final prototype we presented during the final session met our initial expectations of the project, and while internally everything functioned properly the final prototype did prove quite challenging to operate. In our view, this comes down to the large amount of variance between users in regards to how time intervals are judged. During our own testing of our final prototype as well as during the presentation we found that the participant receiving the haptic messages from the sender would often decode the message successfully. The output of the message that would show on the LCD however would often be incorrect at the same time. One of the main takeaways of building this installation has been the complexity of hard-coding events to trigger in Arduino on a time-interval base, and subsequently making these time-intervals as intuitive as possible for participants. As individual participants all have different internal clocks on how to judge concepts like ‘short’ and ‘long’, it takes a lot of practice to actually grow accustomed to the hard-coded time intervals. As such, the only way we envision the difficulty curve for getting the Morse code letters to successfully show up on the LCD screen is to somehow internally code the translation of the biting sequences into output to respond to the input dynamically. This will most likely prove to be quite complicated. Throughout the testing phase we tweaked the parameters for registering ‘short’ or ‘long’ presses as well as the time intervals between letters multiple times, with the goal of finding the most user-friendly setup for the system.

The pressure sensors and vibrating actuators function consistently. When utilizing the longer 0.55mm^2 cables, we noticed that overtime the connection of these wires with the breadboard became slightly loose, resulting in the vibrating actuator not generating output. In the end this proved easily fixable by hand. The mouthpiece and sanitation procedures we crafted, although far from optimal, did the job well and sufficiently protected the pressure sensor while also ensuring users were not exposed to germs from other users. No users made comments about experiencing negative feelings or sensations in regards to the actuator even though this was noted as an issue in the original research paper (Hart et al., 2016). Although prior experience with Morse code can make it easier to decode messages faster, we found that even for those who know little to nothing about Morse code the installation was intuitive enough for participants to catch on quickly.

Licensing

As stated previously we were inspired by the attached research that was aimed at helping medical patients that were unable to communicate through normal verbal means. We had already conceived of a similar signal flow but our project came into more focus as we implemented the contents of the paper. Our project is distinct from that in the research in the sense that our aim was to create an engaging interactive game that challenged users to communicate in a playful and unorthodox way rather than to serve as a tool for some sort of medical treatment. Also we developed this prototype with our own independent research that did not explicitly borrow any code or schematic from an explicitly copyrighted source. Additionally from our research we did not find another version of this apparatus applied to a more playful and game-like context, which leads us to consider this as a genuine innovation within our scope. If we are to make modifications to this in the future we will likely reach out to the original authors of the paper to at least inform them that we are referencing their work in a related project. In its current state this prototype under the provisions of an implied oral contract between the group members (as stated in the authors section of this document), meets the baseline conditions for a future patent application and subsequent patent pending status unless otherwise explicitly stated and annulled in future documentation. The authors state this claim to be valid due to the creation of unique source code, unique hardware and combined and supplementary physical ornamentation that in combination create a unique Intellectual Property entity. This also includes the trademark *'Biting Banter'* as a distinct implied trademark.

Arduino code

```
#include "HD44780_LCD_PCF8574.h"

// LCD Configuration
#define DISPLAY_DELAY_INIT 50 // mS
HD44780LCD myLCD(2, 16, 0x27, &Wire); // Instantiate an object

// FSR and Morse Code Variables
int fsrAnalogPin = 0;
int fsrReading;
```

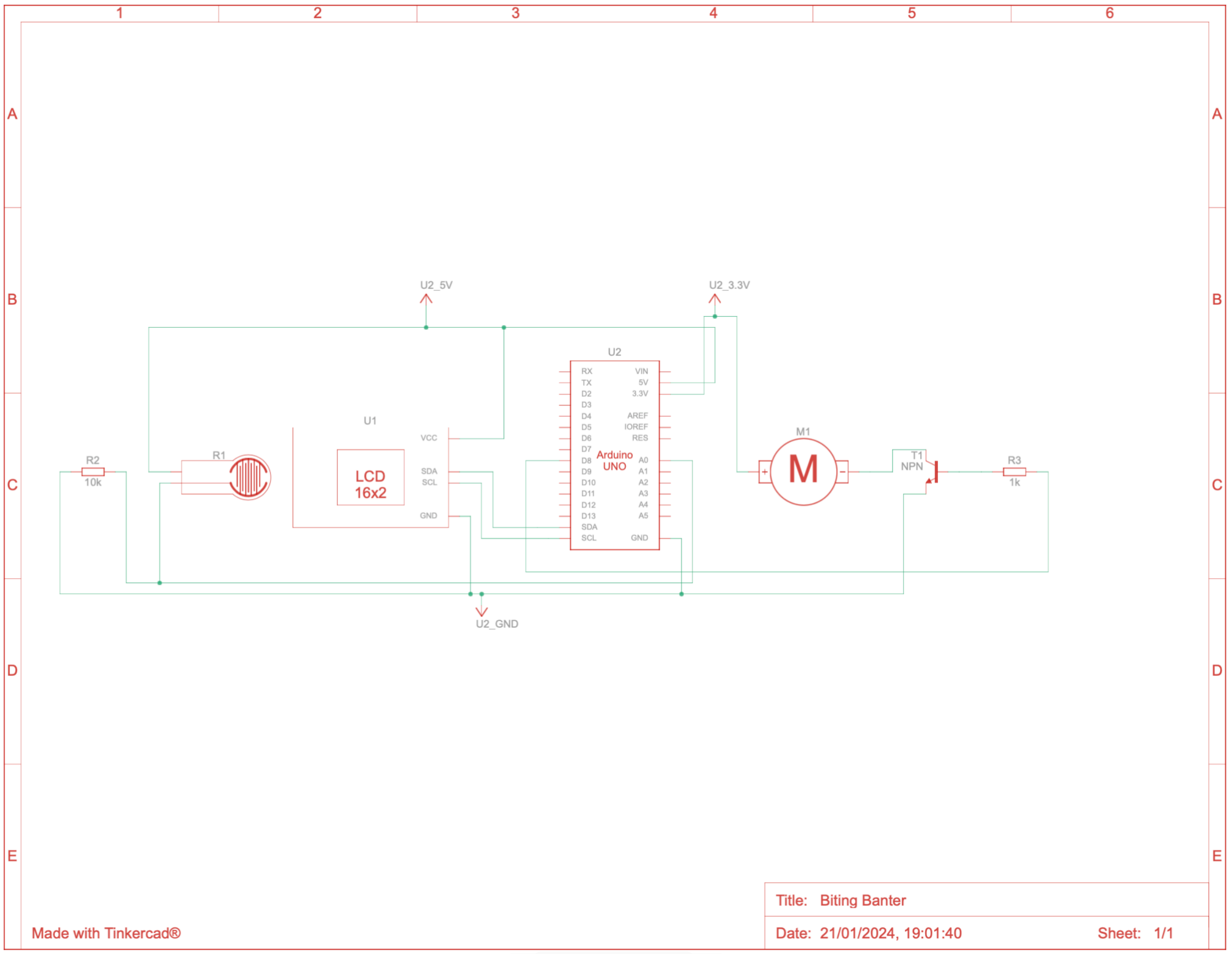
```
unsigned long pressStartTime = 0, pressEndTime = 0, pressDuration = 0,
lastPressEndTime = millis();
bool isPressed = false;
String pressSequence = "";
String displayString = ""; // String to hold the translated Morse code

// Morse Code Mapping
String morseCode[26] = {
 ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---",
 "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-",
 "..-", "...-", ".--", "-..-", "-.--", "--.."
};

void setup() {
 Serial.begin(9600);
 pinMode(8, OUTPUT);

 delay(DISPLAY_DELAY_INIT);
 myLCD.PCF8574_LCDInit(myLCD.LCDCursorTypeOn);
 myLCD.PCF8574_LCDClearScreen();
 myLCD.PCF8574_LCDBackLightSet(true);
 myLCD.PCF8574_LCDGOTO(myLCD.LCDLineNumberOne, 0);
}

String translateMorseCode(String morse) {
 for (int i = 0; i < 26; i++) {
   if (morseCode[i] == morse) {
     return String(char('A' + i));
   }
 }
 return ""; // Return empty string if not found
}

void loop() {
 fsrReading = analogRead(fsrAnalogPin);

 if (fsrReading >= 100) {
   digitalWrite(8, HIGH);

   if (!isPressed) {
     pressStartTime = millis();
     isPressed = true;
```

```
    }
  } else if (isPressed) {
    digitalWrite(8, LOW);

    pressEndTime = millis();
    pressDuration = pressEndTime - pressStartTime;
    isPressed = false;

    if (pressDuration >= 100 && pressDuration < 1000) {
      pressSequence += ".";
    } else if (pressDuration >= 1000 && pressDuration <= 2000) {
      pressSequence += "-";
    } else if (pressDuration > 3000) {
      myLCD.PCF8574_LCDClearScreen();
      displayString = ""; // Clear the display string
      Serial.println("LCD Cleared, Ready for New Input");
      lastPressEndTime = millis(); // Reset the last press end time
      return; // Exit the current loop iteration
    }

    lastPressEndTime = millis();
  }

  if (millis() - lastPressEndTime > 2000 && pressSequence.length() > 0) {
    String letter = translateMorseCode(pressSequence);
    if (letter != "") {
      displayString += letter; // Append the letter to the display string
      myLCD.PCF8574_LCDGOTO(myLCD.LCDLineNumberOne, 0);
      myLCD.PCF8574_LCDSendString(displayString.c_str()); // Display the updated string
      Serial.print("Morse Code: ");
      Serial.print(pressSequence);
      Serial.print(" -> Letter: ");
      Serial.println(letter);
    } else {
      Serial.println("Invalid Morse Code");
    }
    pressSequence = "";
  }

  delay(100);
}
```

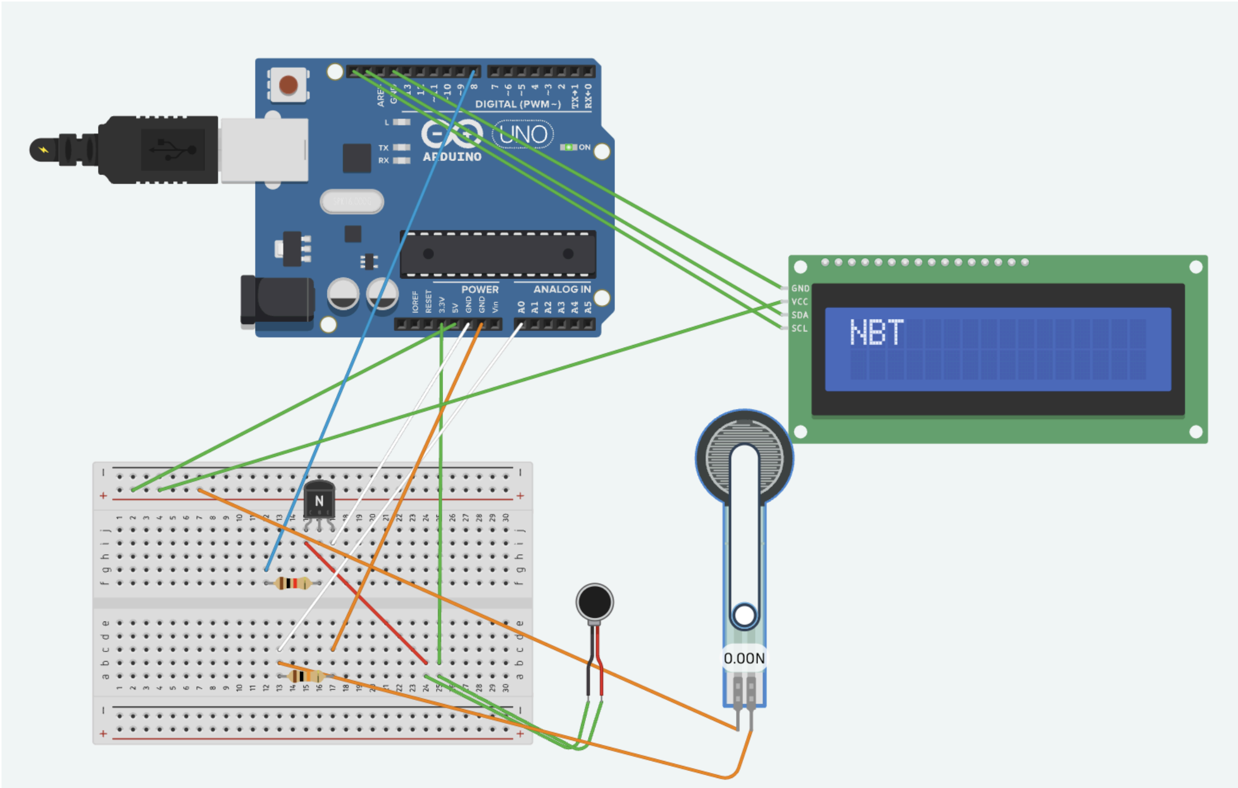
Video of working project

[Google Drive]

Schematic



Name	Quantity	Component
U2	1	Arduino Uno R3
U1	1	PCF8574-based, 32 LCD 16 x 2 (I2C)
R1	1	Force Sensor
M1	1	Vibration Motor
T1	1	NPN Transistor (BJT)
R2	1	10 kΩ Resistor
R3	1	1 kΩ Resistor



Pictures of hardware

[Google Drive]()

Bibliography

1. Hart, K., Chapin, J., & Reed, K. (n.d.). Haptics Morse Code Communication for Deaf and Blind Individuals. *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 2016*. http://reedlab.eng.usf.edu/publications/hart2016morse.pdf