A Morse Input Device for PC Word Processing

Ralph Irons N4RLI

The Design

As described on page 78 of the June 2019 issue of QST, the QEX Morse Input Design Challenge is based on the premise that "the use of a paddle to input text to a personal computer in Morse formal seems like a promising solution to the age-related and disabilities-related keyboard handicap." The Challenge is to create a "Morse key input, along with a control box or other adjunct implementing **SHIFT**, **BACKSPACE**, **ENTER**, **TAB**, and other non-Morse characters."

My design uses a capacitive touch-sensing paddle with input to an Arduino Leonardo to send ASCII characters through a USB PC port to any word processing software active in that PC. Squeezing the paddle (touching both sides simultaneously) for one second cycles through speeds of 5 wpm, 10 wpm, 15 wpm and 20 wpm. No additional controls are needed. Non-printing control characters (like those listed in boldface above) are assigned Morse encodings, and entered with the paddle.

The Arduino Leonardo was chosen for this project because it is recognized by PCs as an HID (Human Interface Device), just like a keyboard or a mouse. As a result, it can easily send ASCII characters directly to word processing software, using the Keyboard library available in the Arduino IDE. That library was a big help in coding this project! Unfortunately, the Keyboard library cannot be used with the very popular Arduino Uno.

A short video showing the device in use is at https://youtu.be/7sGvMXWfPKE.

Construction

Paddles are expensive, so I decided to make one. The paddle arm was made using half of a jumbo craft stick, and touch sensors were attached to each side. I found that the craft stick by itself was not thick enough to prevent the touch sensors from sensing each other. Experimentation revealed that 5/8" spacing was sufficient to prevent sensor interaction. A layer of electrical tape keeps the jumper wires from the sensors in place, and provides the craft stick with enough padding to fit snuggly into the groove of a half-size breadboard. The paddle arm was secured using craft glue (E6000) in the breadboard groove. An LED and a piezo buzzer are included on the breadboard to provide visual and audio feedback while using the paddle.
The breadboard and the Leonardo were both affixed to a cardboard panel which is held in place by the laptop used for Morse input. See Figure 1.

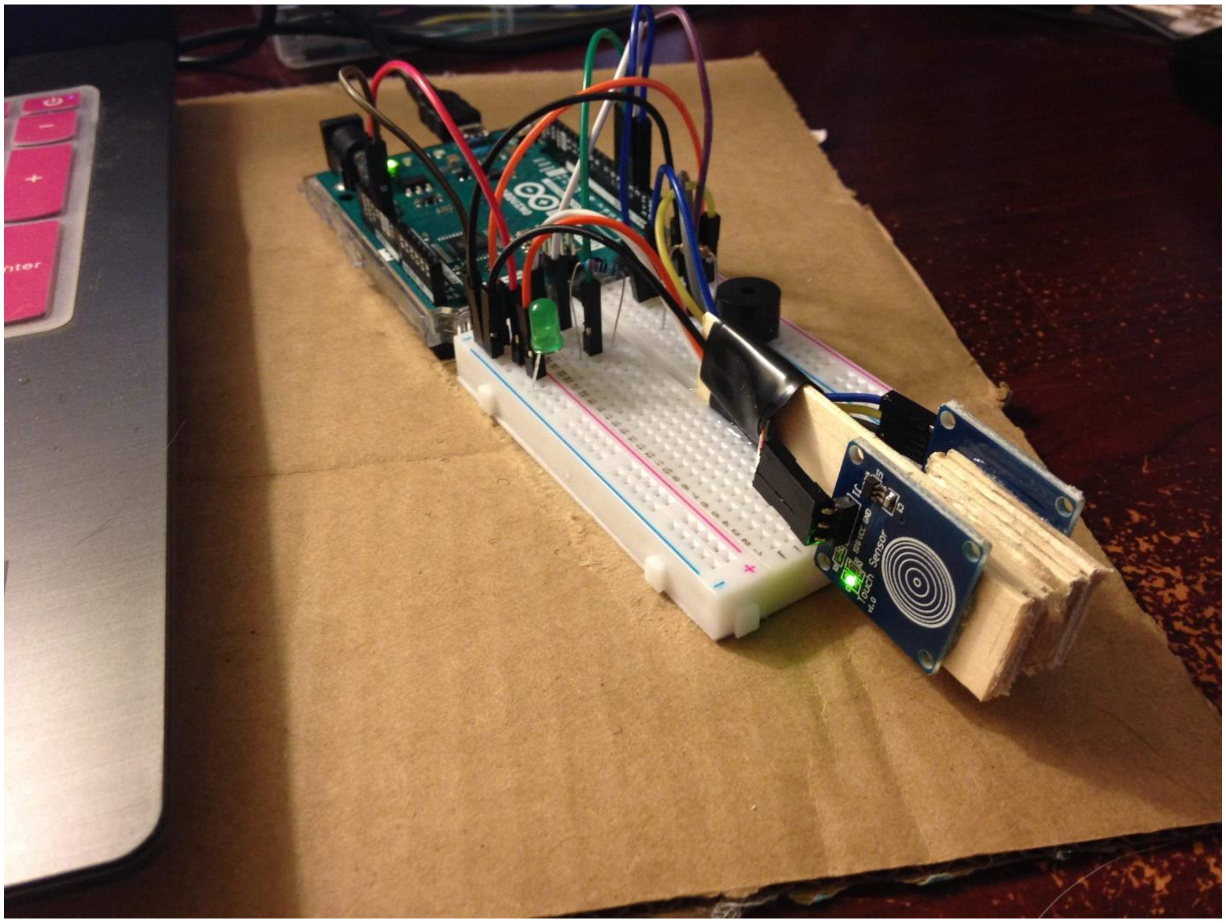

Figure 1. The paddle, breadboard and Leonardo all mounted on a cardboard panel.

Parts List

Item	Source
Arduino Leonardo	https://www.amazon.com/KEYESTUDIO-Leonardo-Development-Board-Arduino/dp/B0786LJQ8K/
Micro USB cable	Comes with the Leonardo clone from the above source
LED	https://www.amazon.com/Gikfun-Assorted-Arduino-100pcs-EK8437/dp/B01ER728F6
Piezo buzzer	https://www.amazon.com/GFORTUN-Terminals-Electronic-Continuous-Industrial/dp/B0716FD838
100 ohm and 220 ohm resistors	https://www.amazon.com/Resistor-Assorted-Resistors-Assortment-Experiments/dp/B07L851T3V
10 cm Dupont Jumpers (6 female-male, 7 male-male)	https://www.amazon.com/EDGELEC-Breadboard-Optional-Assorted-Multicolored/dp/B07GD1XFWV
Half-size breadboard	https://www.amazon.com/Breadboard-Solderless-Prototype-PCB-Board/dp/B077DN2PS1
Touch sensors	https://www.amazon.com/WMYCONGCONG-TTP223B-Digital-Capacitive-Arduino/dp/B07JVGCYGX
Jumbo craft sticks	https://www.amazon.com/Karlash-Jumbo-craft-sticks-length/dp/B072MQTWGH

Wiring Diagram

A Fritzing wiring diagram is given in Figure 2. An ASCII wiring diagram is included in comments in the Arduino sketch, available at http://www.mathmage.org/MorseInput .

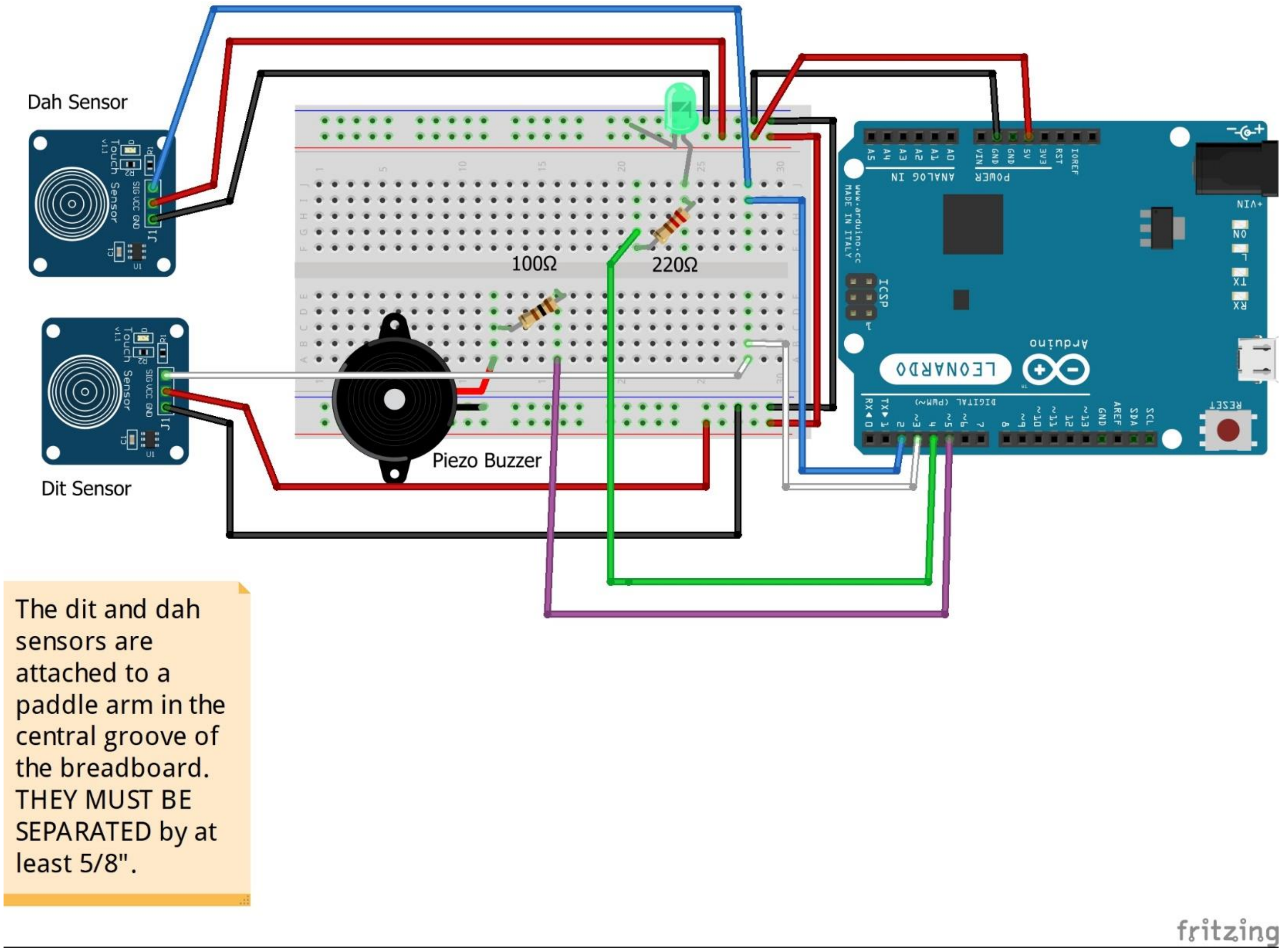
Dah Sensor
Dit Sensor
100Ω
220Ω
Piezo Buzzer
The dit and dah sensors are attached to a paddle arm in the central groove of the breadboard. THEY MUST BE SEPARATED by at least 5/8".
fritzing

Figure 2. A Fritzing wiring diagram of the project.

Overview of Arduino Sketch

Each sensor drives its digital pin HIGH when it senses touch. The main loop of the Arduino sketch waits for the pins to go HIGH, and generates a stream of dits (or dahs) for as long as the pin is HIGH, adding them to a code word as they are produced. Pauses in sensor pressing are also detected, and their durations summed. When the pause is long enough to indicate the completion of a letter, the code word is decoded to ASCII and sent to the PC.

User input is first encoded numerically, using decimal values. A dit in the n^{th} position (starting with $n = 0$) is represented by $1x10^n$, while a dah in the n^{th} position becomes $2x10^n$. As an example, to produce the letter 'u', the user first presses the dit side of the paddle long enough to produce two dits. The first dit is represented by $1x10^0 = 1$, and the second dit is represented by $1x10^1 = 10$. These values are added to the code word, whose initial value is 0. The numerical code word is now 1 + 10 = 11. The user then immediately presses the dah side of the lever long enough to produce one dah, which is represented as 200. The code word is now 11 + 200 =

211. The user now refrains from touching the paddle, and when three dits worth of time has passed, the numerical code word 211 is identified as ASCII ‘u’ in a lookup table. The Keyboard library command **Keyboard.write** is then used to send the letter ‘u’ to the word processor open on the PC desktop.
The standard scheme[1] is used for setting speed. Dahs are equal in duration to three dits, the spacing between dits and dahs in the same letter is one dit, and the pause between letters is three dits in length. Dit duration T for a given speed S (in wpm) is given by

$$T = 1200/S.$$

The above result is only approximate for this Morse input device, since it does not take into account processing time due to inefficient coding (of which I am certainly guilty).
The complete sketch is available for download at http://www.mathmage.org/MorseInput .

Uploading the Sketch

Be aware that once the Arduino sketch for this project is uploaded to a Leonardo, it will be able to type into any active word processing program on your PC desktop – including the Arduino IDE itself! If you touch the paddle after uploading the sketch, the resulting ‘t’ or ‘e’ will be typed directly into the Arduino sketch, if it is still open on your desktop.
To prevent corruption of the code, I move the cursor to a comment section before clicking the ‘upload’ arrow in the Arduino IDE.

Operation

The International Morse Code[2] contains encoding for much punctuation which I have never heard on the air as a ham radio operator. Yet those symbols (like colons, exclamation points, double quotes or parentheses) are widely used in word processing, so they were included in the lookup table. Several characters not incorporated in the International Morse Code are also essential to word processing. Some are non-symbolic control characters (like Space, Backspace, Tab, Enter and Shift), while others are symbolic (like $ or @). These were assigned encodings not used in the standardized International Morse Code, and added to the lookup table.

A complete table of the encodings used in this project is in Figure 3. The user will want to have a copy of this chart for reference while operating the Morse input device.

Control Characters	**Morse**		**Alphabet**	**Morse**	**Alphabet**	**Morse**
Shift	..-.-		a	.-		
Space	.-.-		b	-...	**Digits**	
BackSpace	.-.--		c	-.-.	0	-----
Enter	.-..-.-		d	-..	1	.----
Tab	--...--		e	.	2	..---
Esc	...-.-.		f	..-.	3	...--
Up Arrow	..-.--.		g	--.	4	....-
Down Arrow	-...--		h	....	5	.....
Left Arrow	.-..-		i	..	6	-....
Right Arrow	.-.--.		j	.---	7	--...
Punctuation			k	-.-	8	---..
Ampersand &	.-...		l	.-..	9	----.
Apostrophe ‘	.----.		m	--	**Extras**	
At sign @	.--.-.		n	-.	%	.--.-.-.
Left parenthesis (	-.--.		o	---	#	....--.
Right parenthesis)	-.--.-		p	.--.	^	-.-.-
Colon :	---...		q	--.-		
Semicolon ;	-.-.-.		r	.-.		
Comma ,	--..--		s	...		
Equals =	-...-		t	-		
Exclamation point !	-.-.--		u	..-		
Period .	.-.-.-		v	...-		
Hyphen -	-....-		w	.--		
Plus +	.-.-.		x	-..-		
Quest mark ?	..--..		y	-.--		
Quotes “	.-..-.		z	--..		
Slash /	-..-.					
Dollar sign $	...-..-					

Figure 3. A Table of all the encodings used in this project. Green represents encodings not in the ITU-R recommendations, but which seem to be in use. Pink represents non-ITU-R encodings invented for this project.

I'd recommend that the user start at the lowest speed of 5 wpm. Lightly touch only one side of the paddle at a time. Try not to leave pauses between sending dits and dahs for the same letter. Leaving pauses (more specifically, three dits worth or more) invites the Arduino sketch to conclude that you have finished sending a letter.

When the Arduino finishes decoding a letter, it will transmit it to your PC. You will see the TX LED (near the micro USB cable jack) flash. Do not begin sending another letter until after this flash has occurred, or the letter has appeared on the PC screen.

To increase speed, lightly squeeze the paddle (touching both sides) for one second. The LED and piezo buzzer will announce the new speed in Morse code. Repeating this will cycle through speeds of 10 wpm, 15 wpm, 20 wpm, then back to 5 wpm.

Notes

[1] See "International Morse Code", Recommendation ITU-R M.1677-1, October, 2009, downloadable at https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1677-1-200910-I!!PDF-E.pdf , p. 3.

[2] See "International Morse Code", Recommendation ITU-R M.1677-1, October, 2009, downloadable at https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1677-1-200910-I!!PDF-E.pdf , p. 1.