

Tweet with Morse code



PJ Evans

MAKER

PJ is a writer and software engineer who seems to want to make communicating with the outside world as challenging as possible.

@mrpjvans

If your keyboard has you bored, why not learn Morse and then be able to send tweets using nothing but a simple switch?

When Samuel Morse created his communication system in the mid-1800s, it revolutionised wireless communication. The original idea was that simple electronic pulses could be sent further and more reliably than voice, and so the Morse alphabet was used to describe each letter and number as a combination of short pulses (dots) and long pulses (dashes).

Learning Morse code is a challenge, but can be rewarding and a lot of fun. As an introduction to this classic way of communicating, we're going to build a tweeting Morse code key. We'll learn how to interpret input and also how you can create more complicated projects by breaking them down into small pieces.

You'll Need

- ▶ Small breadboard magpi.cc/minibreadboard
- ▶ Tactile switch magpi.cc/switches
- ▶ Adafruit 16x2 LCD Keypad Kit (optional) magpi.cc/keypadkit
- ▶ Morse key (optional) magpi.cc/morsekey
- ▶ Twitter account (optional) twitter.com
- ▶ Jumper wires

▶ This popular display means we can see our Morse code without the need for a monitor

01 Let's get set up

First, select the right Raspberry Pi model for the job. Of course, we would heartily recommend a Raspberry Pi 4, but in fact this project will not be too demanding on even the oldest models, so it is great for upcycling an older Raspberry Pi computer. In fact, it will even work with the original Model A and B.

Start by installing the latest version of Raspbian. We've no need for a graphical user interface, so you can use Raspbian Lite if you wish; whatever is most comfortable. We'll be doing everything in the command line.



02 Configure and update

This project has several steps, so don't worry if you just want to practise Morse code – we'll get to that first. If you want to complete everything here, you'll need to set up an internet connection (wireless or wired) and enable I²C, which is used to communicate with the LCD screen. By running `sudo raspi-config` from the command line, you can enable WiFi under 'Network Options' and I²C under 'Interfacing Options'.

Whatever it is that you've decided to do, always make sure you've updated the system by running `sudo apt update && sudo apt upgrade`. This may take some time; once complete, it's important you reboot so that I²C is properly enabled.

03 Get switched on

Let's try to emulate the Morse key by using a tactile switch. These widely available and inexpensive switches make a satisfying 'click' when pressed (hence the name). They have four pins – two pairs that are connected on the longer side, so the switching is done between those with the shorter gap. Bearing this in mind, place the switch into the breadboard so the longer edge follows the connected rows. Don't worry if you make a mistake: nothing can be damaged. Now connect the breadboard to your Raspberry Pi's GPIO. Run jumper leads on each side of the switch to the last two pins at the end (nearest the USB ports) of the GPIO header: GND and GPIO 21.

04 Coding time

Once you've checked all your connections, create a new file called `morse.py` in your favourite editor and enter the code listed here. You can also download it from magpi.cc/twitterkey if you don't fancy the typing. The code will listen for changes to the button's 'state' (whether it is pressed or not) and measure the time differences



This 16x2 LCD display shows us the output from our practice

This training key can be added to create an authentic Morse code experience

to work out whether you made a 'dot' or a 'dash'. It will then convert the pattern into a letter and display it on the screen.

First, install these dependencies (libraries that help us):

```
sudo apt install python3-pip
pip3 install gpiozero
```

Then run `python3 morse.py`.

05 Practise dots and dashes

Using the chart printed here (**Figure 1**), see if you can spell your name out by clicking the button. Use a quick press for a 'dot' and a slightly longer press for a 'dash'. Leave the button untouched for a slightly longer time to tell the code you've finished your letter. Once you're happy everything is working and you've had some fun, **CTRL+C** will stop the program.

If you're not happy with the timings, you can adjust them to suit your 'fist' (the name operators give their style of keying). You can adjust the timings for a dot, dash, and interval between letters by changing the timings in the variables `dot_timeout` and `dash_timeout` at the start of the code. Don't be afraid to experiment.

Figure 1

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	• —	U	• • —
B	— • • •	V	• • • —
C	— • — •	W	— • —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — •		
H	• • • •		
I	• •		
J	• — — —		
K	— • —	1	• — — — —
L	• — • •	2	• • — — —
M	— —	3	• • • — —
N	— •	4	• • • • —
O	— — —	5	• • • • •
P	• — — •	6	— • • • •
Q	— — • —	7	— — — • •
R	• — • •	8	— — — — • •
S	• • •	9	— — — — •
T	—	0	— — — — —

▲ **Figure 1** The Morse code alphabet. Although it appears random, there is an underlying structure that helps you understand and memorise the patterns

Top Tip

Learning Morse

There are many resources online that will help you understand how Morse code is structured. Start with learnmorsecode.info

06 Build the LCD

To allow us to create Morse code without the need for a full display, we've selected a bright, crisp and slightly-retro LCD screen from Adafruit. These popular panels normally require a lot of GPIO pins to be driven natively, but this HAT uses an input extender so that only two pins are needed. Better yet, it comes with five tactile switches on-board so we can use them for input.

This LCD kit arrives unassembled, so it's time to get the soldering iron out. There are excellent assembly instructions at magpi.cc/lcdkitinfo. As always, read through them before doing anything and take your time.

07 Set up the display

Once your display is assembled, you can attach it to your Raspberry Pi (make sure the latter is switched off!). Due to the close proximity of some of the resistors, cover the top of the USB ports and Ethernet port with insulation tape if they are close to the PCB of the display.

Test the display is working by installing its Python libraries:

```
sudo pip3 install adafruit-circuitpython-charlcd
```

Now create a new file called `lcd.py` and enter the code from the listing here. Save it and run it using `python3 lcd.py`. The display should show your message. If it lights up but you can't see anything, adjust the 'Contrast' potentiometer until the text appears.

lcd.py

► Language: **Python 3**

```
001. import board
002. import busio
003. import adafruit_character_lcd.character_lcd_rgb_i2c as
    character_lcd
004. lcd_columns = 16
005. lcd_rows = 2
006. i2c = busio.I2C(board.SCL, board.SDA)
007. lcd = character_lcd.Character_LCD_RGB_I2C(i2c,
    lcd_columns, lcd_rows)
008. lcd.color = [100, 0, 0]
009. lcd.message = "Hello\nFrom MagPi"
```

08 Version 2

It's time for a more advanced version of our original code. This one is a bit longer, so download `lcd_morse.py` from magpi.cc/twitterkey. This time we're reading input from the LCD's on-board tactile keys, so the code needs to be a bit different. The time measurement variables are still there. Run it using `python3 lcd_morse.py`.

You should be able to key away and see the interpreted letters appear on screen. You now have a functioning standalone Morse code trainer.

09 Let's tweet

We'd like to be able to send our messages to Twitter. For security reasons, we need to create a Twitter 'application' which gives the code unique credentials for posting on our behalf. We're using the python-twitter library – see magpi.cc/pytwitterinfo for an excellent tutorial on how to set it up. You will be given four strings: a consumer key, consumer secret, access token, and access token secret. Enter all the values in the equivalent variables in the first few lines of `lcd_morse_twitter.py` (download the code from magpi.cc/twitterkey). Now save the file.

10 Tweet with Morse!

Run `python3 lcd_morse_twitter.py`. As before, you can construct your message by tapping on the right-hand cursor button of the LCD display. Your message will be displayed at the top, and the current dots and dashes in the 'buffer' at the bottom. Made a mistake? No problem: click the left-hand cursor to delete the previous character or the 'up' key to delete the entire message and start again. When you're happy, click on 'Select' to send. Your message will be posted to your account for all of Twitter to read.

11 Add a Morse key

Let's take the authenticity up a notch by adding a real Morse key. These keys are nothing more than a simple on/off switch. That said, some can be surprisingly expensive as they are built using precision components to allow the operator to go faster and faster with fewer mistakes. We've selected a more affordable training key that has two contacts that can be directly connected. To use the

morse.py

► Language: Python 3

DOWNLOAD
THE FULL CODE:

 magpi.cc/twitterkey

```


001. from gpiozero import Button
002. import time
003.
004. button = Button(21) # GPIO Pin 40
005. dot_timeout = 0.15
006. dash_timeout = 1
007. current_letter = ""
008.
009. morse = {
010.     ".-": "A", "-...": "B", "-.-.-": "C", "-.-": "D",
011.     ".": "E",
012.     "-.-.-": "F", "-.-": "G", "....": "H", "..": "I",
013.     ".---": "J",
014.     "-.-": "K", "-.-.-": "L", "--": "M", "-.-": "N",
015.     "----": "O",
016.     "-.-.-": "P", "-.-.-": "Q", "-.-": "R", "...":
017.     "S", "-": "T",
018.     ".-.-": "U", "-.-.-": "V", "-.-": "W", "-.-.-":
019.     "X", "-.-.-": "Y",
020.     "-.-.-": "Z", "....": "1", "....": "2",
021.     "....": "3", "....": "4",
022.     ".....": "5", ".....": "6", ".....": "7",
023.     ".....": "8", ".....": "9",
024.     "-----": "0"
025. }
026.
027. while True:
028.     # Wait for a keypress or until a letter has been
029.     # completed
030.     button.wait_for_press(dash_timeout)
031.
032.     # If we've timed out and there's been previous
033.     # keypresses, show the letter
034.     if button.is_pressed is False and
035.     len(current_letter) > 0:
036.         print("\nMorse: " + current_letter)
037.         if current_letter in morse:
038.             print("Letter: " + morse[current_letter])
039.         else:
040.             print("Not recognised")
041.         current_letter = ""
042.
043.     elif button.is_pressed:
044.         # The key has been pressed, work out if it's
045.         # a dot or a dash
046.         button_down_time = time.time()
047.         button.wait_for_release()
048.         button_up_time = time.time()
049.         button_down_length = button_up_time -
050.         button_down_time
051.
052.         # Was it a dot or dash?
053.         if button_down_length > dot_timeout:
054.             print('-', end='', flush=True)
055.             current_letter += '-'
056.         else:
057.             print('.', end='', flush=True)
058.             current_letter += '.'
059.
060.         time.sleep(0.1)

```

existing code, solder two wires to the underneath of the rightmost tactile switch on the LCD board and connect them to the key using its screw terminals. Now you can key away using the real thing!

12 Going further

Now you have the basics as Python code, you can repurpose your tweeting Morse key for anything you can imagine. Add a second key and create Morse code challenge games. How about Morse code hangman? Add timing in to see how many letters per minute you can key. Could two identical setups send messages to each other?

Although initially challenging, learning Morse code is rewarding and can inspire operators to go on to the rich and fascinating world of amateur radio. Over to you. 

Top Tip



Make a key

If you don't fancy the expense of buying a Morse key, you can make your own! magpi.cc/diymorsekey

