ESP32 PORTABLE KEYER BY EAHVO

There are hundreds of neat keyer designs on internet so, why designing another one?

At the beginning it was only a personal challenge. Eventually I like the result. This is the reason I share this desing.
It is smaller than the smaller ones, it is very cheap and easy to build, it is really flexible and strong enough to undergo the hardest conditions.
It has no buttons or potentiometers, so it is really simple to build. The box is 3d printed.
Power supply is 5v USB so it can be powered by a usb power bank. Suitable for portable operation.

Some of its features:

1)-Bluetooth connection. It allows control from mobile phone and other devices.
2)-Up to 25 pre-saved messages (even more, depending of the chosen app) easily written using natural language
 Messages are translated automatically to Morse code.
3)-Variable speed from 5 to 70 wpm.
4)-Suitable for paddle key or straight key.
4)-Reverse mode for left-handed operators.
5)-Internal beeper for learning purposes.
6)-Every command is played in Morse code (for blind operators).
7)-It has no push buttons or potentiometers.
8)-3 stored messages in standalone operation. Up to 30 character per message.
9)-Every message could be repeated automatically in a time from 5 to 9999 seg. Repetition stops when touching the paddle/key. Very useful for CQ calling.
10)-Repetition continues after a power supply cut. Useful for beacon operation.
11)-Menu in standalone operation accessible via touch buttons.
12)-Speed control via touch buttons. It plays the final speed in Morse code.
13)-Adjustable touch sensitivity.
14)-Adjustable paddle delay sample from 1 to 99.
15)-Tune function. Transmit a continuous carrier for 10 second to adjust SWR.

Different types of messages:

This keyers translates text to Morse code so you only have to write the messages in your own language. Three different types of messages are possible.

Message without repetition. –

This message is used for transmitting once every time you push the button. The structure is:
mescq cq cq de ea7hvo#
mes is the header used for message without repetition
cq cq cq de ea7hvo is the message to transmit in cw . Up to 30 characters per message are admitted.
# is the end of file character.

Message with repetition. –

repcq cq cq de ea7hvo0005#
rep is the header used for message with repetition
cq cq cq de ea7hvo is the message to transmit in cw . Up to 30 characters per message are admitted.
0005 is the time in seconds between repetitions. 9999 seconds are possible.
# is the end of file character.
The keyer stores the message and the time so it continues transmitting after a power supply problem.
The repetition stops when:
-1) Touching left or right paddle. You can call CQ until someone answer you and, then repetition stops when you touch the paddle.
-2) Touching a touch button.
-3) Sending a message without repetition

Message stored in touch buttons 1, 2 and 3. –

st1cq cq cq de ea7hvo0005# stores message in touch button 1
st2qth is cordoba0005# stores message in touch button 2
st373 de ea7hvo0005# stores message in touch button 3
st1/st2/st3 is the header used to store messages 1, 2 and 3
cq cq cq de ea7hvo is the message to transmit in cw . Up to 30 characters per message are admitted.
0005 is the time in seconds between repetitions. 9999 seconds are possible. 0000 seconds implies no repetition in the stored message.
# is the end of file character.
Once you have stored these messages, you can send them without using your mobile phone.

Following the same structure used for messages, the keyer admits several commands :

1)-TUNE FOR SWR ADJUST-

this command switches ON/OFF TX .A continuous carrier is transmitted for 10 second.
It stops touching the paddle or sending another command.
You can stop it or activate it sending this string from your mobile: tun#

2)-SOUND ON/OF.-

this command switches ON/OFF the internal beeper . Intenal beeper is useful in diy project and qrp ones.
You can stop it or activate it sending this string from your mobile: sou#

3)-LED ON/OFF

This command switches the internal esp32 led.
You can stop it or activate it sending this string from your mobile : led#

4)-MODE LEARNING/OPERATE

This command stops the control over the transceiver and activates the beeper.
You can stop it or activate it sending this string from your mobile : lrn#

5)-MODE STRAIGHT KEY/PADDLE

If you wanted to use a straight key you could do it without losing the stored messages. In this case, the iambic keyer is desactivated but not the momories (touch memories and mobile memories).
Sometimes it is neccesary a reverse command when using straight key. Depends on the wiring of your jack.
You can activate this fuction sending this string from your mobile : str#

6)-REVERSE PADDLES FOR LEFTHANDED PEOPLE

this command switches the sign sent by every paddle. Dits and dashes are sent by a different paddle. It is useful for lefthanded people.
You can activate this fuction sending this string from your mobile : rev#

7)- FACTORY RESET FOR FAILSAFE PREFERENCES

you can restore failsafe preferences only sending this string : fct#.
It doesn't erase the messages stored in touchbuttons 1 to 3.
It affect to Speed, touch sensitivity, reverse, straight key, sound and led functions.

8)-PADDLE DELAY SAMPLE

Paddle delay sample is controlled by a seekbar but you can also control it by strings.
The string that controls Paddle delay is for instance : pds052 this command sets the delay in 52

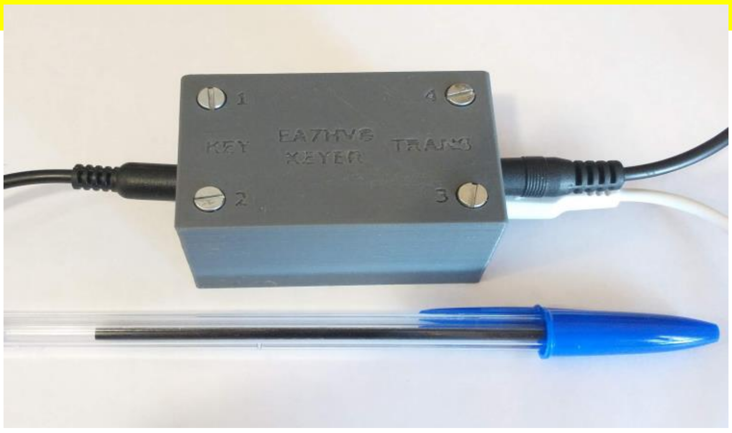
9)-SPEED

Speed is controlled by a Seekbar but you can also control it with strings.
The string that controls speed is for instance : vel032 this command set spedd in 32WPM.
The seekbar is explained in the bluetooth instructions
This keyer is limited to 70WPM max.

There are several app that send serial messages over bluetooth. My favorite is this:
https://play.google.com/store/apps/details?id=nextprototypes.BTSerialController&hl=es_419&gl=US

I have prepared a guide to configure this app . It is included in the documentation.

CONNECTIONS



There is a side labeled KEY where you have to insert the jack of the paddle/key
The side labeled TRANS is for the transceiver jack.
In the same side you have the microUSB connector for the power suply.

TOUCH BUTTONS AND MENU

The four screws labeled with numbers from 1 to 4 are also touchbuttons.
Buttons 1 to 3 send stored messages.
To store messages send via bluetooth :
st1cq cq cq de ea7hvo0005# stores message in touch button 1
st2qth is cordoba0005# stores message in touch button 2
st373 de ea7hvo0005# stores message in touch button 3
st1/st2/st3 is the header used to store messages 1, 2 and 3
cq cq cq de ea7hvo is the message to transmit in cw . Up to 30 characters per message are admitted.
0005 is the time in seconds between repetitions. 9999 seconds are possible. 0000 seconds implies no repetition in the stored message.
# is the end of file character.

SPEED CONTROL:

To increase the speed push firstly button 4 and, simultaneously, button 2.Every wpm increased is shown with a short beep. After a second without touching , the speed is played in morse code.
To decrease the speed push firstly button 4 and, simultaneously, button 1.Every wpm decreased is shown with a short beep. After a second without touching , the speed is played in morse code.

TOUCHMENU

To access to the menu push firstly button 4 and, simultaneously, button 3. A short beepp is played indicating that the menu is started.
Button 3 changes the menu sequentially.
Buttons 1 and 2 select the option in the selected menu.
If we don't touch any button , the keyers will play "sou" in morse code. This is the firts step of the menu.
There are 8 different steps in the menu :

1)-tune.	"tun" is played. Touching buttons 1 or 2 TX is on for 10 seconds to adjust SWR.It stops with a second touch or paddle.
2)- sound .	"sou" is played . Touching buttons 1 or 2 we shitch the sound beeper between on and off.
3)-learning.	"lrn" is player. Touching buttons 1 or 2 we switch the keyer between learning and operate.
4)-straight.	"str" is player. Touching buttons 1 or 2 we switch the keyer between straight key and paddle .
5)-reverse.	"rev" is player. Touching buttons 1 ot 2 the paddles are inverted. Useful for lefthanded people.
6)-led.	"led" is played. Touching buttons 1 ot 2 the visual indication led is turned on/off.
7)-factory.	"factory" is played when touching buttons 1 or 2.Factory reset is called and the keyer restarts
8)-Paddle delay	"pds" is played. Button 1 reduces pds 1 unit every time and button 2 increases 1 unit every time . After a second, the Padle delay sample is played

To finish menu touch the key/paddle or wait 30 second.

```
//**********************************************************
//acknowledgement******************************************
//**********************************************************
//This software is based in
//* Arduino iambic CW keyer v2.0
//* for Ham radio usage
//* Richard Chapman
//* KC4IFB
//* February, 2009
//* Richard Chapman KC4IFB published an article in the Sep/Oct 2009 QEX magazine */
// ********************************************************
//  Iambic Morse Code Keyer Sketch with added Side-Tone
//  added Message keyer and enable poteniometer speed control
//  Copyright (c) Bob Anding 2017
//
//  This library is free software; you can redistribute it and/or
//  modify it under the terms of the GNU Lesser General Public
//  License as published by the Free Software Foundation; either
//  version 2.1 of the License, or (at your option) any later version.
//
//  This library is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
//  Lesser General Public License for more details:
//
//  Free Software Foundation, Inc., 59 Temple Place, Suite 330,
//  Boston, MA  02111-1307  USA
//  ROBERT D ANDING, II (AA5OY) added some modifications you can see in his page:
//  https://bandingsite.wordpress.com/arduino-iambic-keyer/
//  I downloaded the code from his page:
//  https://bandingsite.files.wordpress.com/2017/08/iambic-keyer.odt
//  I have used the keyer and the stored message code as a kernel for my software
//**********************************************************
//
//
```