Accu-Arduino Keyer

Manual

Nick Kennedy, WA5BDU

Introduction

The Accu-Keyer first appeared in QST in the August 1973 issue. The keyer was designed by James Garrett, WB4VVF. It was very popular with CW operators and Mr. Garrett reported having sold 4,000 boards in its first two years and it appeared in several editions of the ARRL Handbook. Around fifty years later, I've become somewhat obsessed with the Accu-Keyer, building one from scratch on a solderless breadboard and refurbishing another which I found at a hamfest flea market. Eventually I got the idea of programming the logic of the Accu-Keyer into an Arduino. I don't mean writing a keyer program that attempts to emulate the Accu-Keyer's characteristics, but to write a program which performs the logic of each gate and flip-flop in the circuit.

Implementation

The logic of the TTL AND and OR gates is unclocked, so a fast loop is used to approximate wired signal propagation with no delays. Flip-flops *are* clocked and so require special logic to emulate. Likewise, the clock itself is emulated using features of the MCU. Paddle inputs and keyer output are easily handled with the I/O lines of the Arduino, and ADC inputs are used for the speed pot and battery monitoring. Also, I/O pins are used to drive a small sidetone speaker and an LED, if desired.

Added features

I wanted to provide some of the useful features one can implement with a microprocessor, while not adding anything to the core logic of the keyer which might change its fundamental nature. I also chose to avoid adding "function buttons", so I'm limited to doing what can be done with the paddle during startup. I chose not to emulate the Accu-Memory. My keyer is battery powered so I added two features with that in mind. First is an alert beep when the keyer is idle for ten minutes (adjustable by user) as a reminder to not leave the keyer on and deplete the battery. Also, the battery voltage is monitored by ADC and checked at startup and each one minute of inactivity. If the voltage is low, an 'alert' beep is sent, plus the letter 'B' for battery. Obviously, I've also added the capability of the keyer to send stored text. It is used sparingly, but a startup message can be sent, plus battery voltage and keyer speed can be reported in Morse.

Original features

The Automatic Character Space (ACS) feature has been incorporated. A switch may be used to turn it off and on, or the user may simply hard wire his preferred choice.

The Accu-Keyer is a mode B iambic keyer.

A manual (hand key) input is also available. In addition, the program can be set to a manual mode in which the paddles may be used to send manual code.

Start-up controls

These features are mutually exclusive – you can only use one at a time. You select a feature by having the paddles in the prescribed state when starting up or pressing reset.

- Both paddles closed: Reports the battery voltage and the current speed in WPM, plus the software version.
- Left (dot) paddle closed: Enter straight key mode. Either paddle gives key down. Hold both paddles to exit straight key mode. If you provide a separate jack from D7 for a straight key input, you could have both straight key and paddle active at the same time.
- Right (dash) paddle closed: Mute the internal speaker. The rig is keyed but there's no internal sidetone.

Hardware options

<u>Power Source</u>

I chose to go with an internal battery. Specifically I'm using a 18650 cell. It puts out about 3.6 volts and will power the keyer for a long time. Since the Arduino requires 5 VDC, I use a boost regulator, which I found to be cheap on eBay.

Since I'm using a battery, I incorporated several features help me to remember to turn the keyer off when not in use and to not run down the battery. First, I have an LED to show that power is on. Second, when the keyer has been idle for five minutes, it issues an alert beep of three ascending tones and repeats it for each additional five minutes of idle time.

Third, the keyer can announce the battery voltage (in Morse) on startup (hold both paddles closed), so the user can determine when the battery needs charging.

Fourth, the keyer monitors battery voltage while in operation and will issue a warning beep plus the letter 'B' in Morse if the voltage gets below a set value.

Note that the user could choose to use a battery or external supply of 8 to 12 V and connect it to the Raw or Vin pin of the Arduino. In this case, the Arduino will be using its own on-board 5 V regulator.

<u>LED</u>

I have an LED and dropping resistor connected between D13 and ground. The LED has two modes and switches between them depending on whether the keyer is idle or is being used.

When the keyer is idle, the LED operates to show that it is turned ON, so it will be on when the power switch is ON.

When the user is sending, the LED is ON for key down and OFF for key up. One minute after the user stops sending, the LED goes back to its power indication mode and lights up.

<u>ACS Switch</u>

This switch allows the user to turn the Automatic Character Space feature On and Off. If this isn't important to you, you can save a little complexity and space by leaving it out and wiring ACS permanently ON or OFF as you choose. Note that the switch is <u>closed</u> for ACS ON. This is the opposite of the switch usage in the hardware version.

Speaker

I used a little (0.5” diameter) sounder for a speaker. I measured 15 Ω resistance. That would pull 333 ma from the 5 V peak square wave - far too much. So I put a 0.39 uF capacitor in series with it to limit current to about 8 mA, and also keep the volume reasonable. You could also use a resistor. There’s a startup option to mute the speaker when desired. You can specify a different pitch by editing the source code.

Speed pot

The value isn’t critical. Something from 5 kΩ to 20 kΩ would be good. The keyer will operate at the default speed if there’s no pot. Ground the A1 pin if you do this. Source code values set the minimum and maximum speeds. You can edit this if you wish. The minimum possible speed is about 8 WPM.

Output circuit

This is the output that keys your transmitter. Note that in modern* transmitters, the key jack presents a low voltage (3.3, 5.0, 12 typical) that you pull to ground in order to key the transmitter. The current sourced is generally quite low: 1 mA or less. I elected to use an opto-isolator to protect my keyer from the scary world. You could just use an interposing transistor (2N2222) or probably even go directly to the Arduino’s pin.

Since the output must pull a voltage to ground, the Arduino must output HIGH for key down, to turn the transistor or opto-isolator ON. But for direct connection of the transmitter’s keyed line, a LOW condition is needed for key down. This keyer gives you one line (D4) as the normal output (HIGH for key down) and another line (D6) that’s the inverse of the D4 line.

* “Modern” is a bit of a stretch. Most rigs built in the last 40 years or so (solid-state designs) key this way.

A minimalist build?

Maybe you just want to try it out first, with a minimal investment in time. You could omit the LED, speaker, ACS switch, and speed pot and the keyer will still function. If you like it, add the hardware you want. You might even power it from a USB connection.

Customizing the source code a bit

There are a number of values in the source code that could be edited to the user’s liking. An easy way to find them is to search for the word: ‘USERS’, which I’ve used to mark such values. Here are some of the things you might want to change:

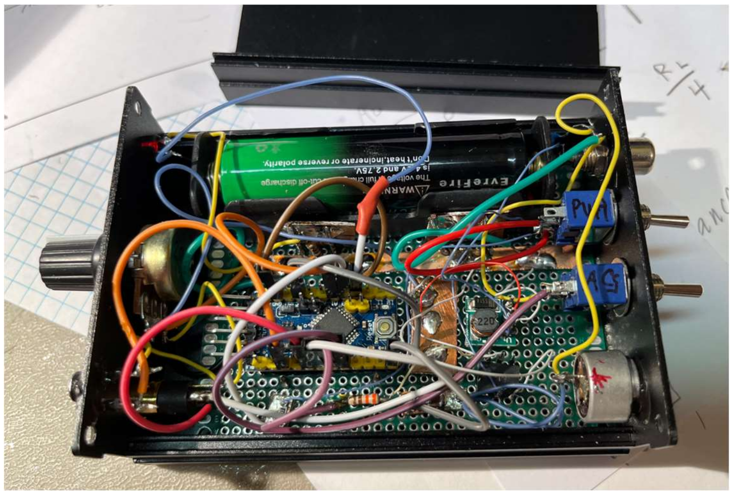
#define pitch 550	Sidetone pitch in Hz
#define WPM 20	Default speed for when no speed pot is installed
#define MSG_WPM 20	Speed used for messages from the keyer
#define ADC_REF 5.05	This should be the voltage on your +5 V pin
#define BATT_LO_ALM 3.0	Battery voltage where the low battery alarm should activate
#define SPD_LIMIT_LO 10.0	Speed when the pot is full CCW (minimum speed)
#define SPD_LIMIT_HI 40.0	Speed when the pot is full CW (maximum speed)

#define UsePot true	Change to 'false' if speed pot is not used
#define IDLE_MS 300000	Milliseconds idle before reminder beep sounds. This is 5 minutes.
bool idle_alert = true;	Change to 'false' if you don't want the idle beep (battery save reminder)
char greeting[] = "EE*";	Start-up message. Consider TU, HI, OK, QRV, R, EE (dit-dit) or maybe your call, or nothing at all. Do keep the * to mark the end.

An enclosure?

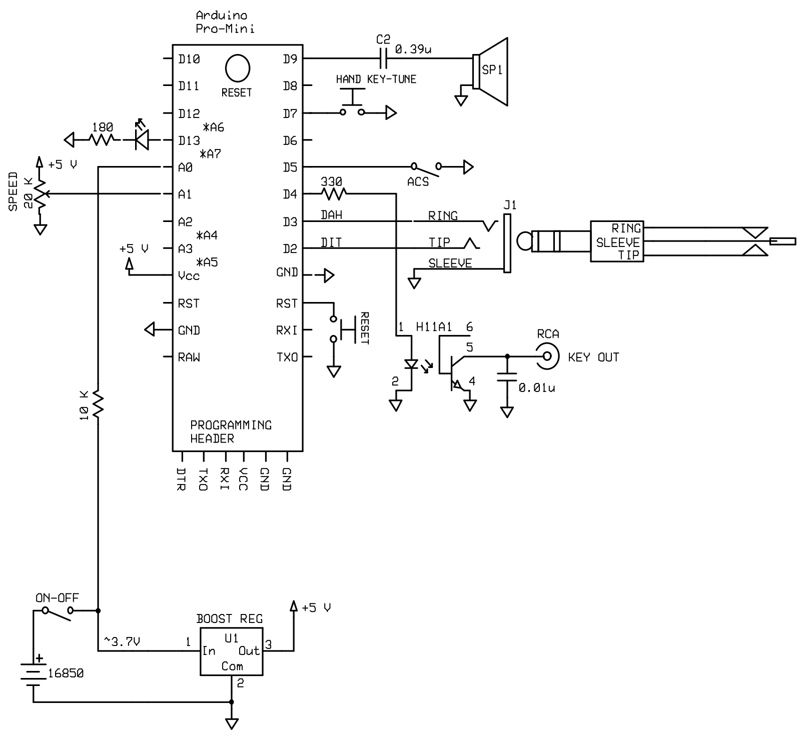
It's up to you. I used an extruded aluminum box with external dimensions of 102 x 76 x 36 mm or 4.05 x 2.95 x 1.4 inches.

Here's a look with the top off:



Speed, LED, Reset, and Paddle in front. Key Out, Power switch, ACS switch, and speaker in the rear.

And here it is in the case:



1) This keyer simulates the logic of the Accu-Keyer from QST AUG-1973

2) Arduino Pro-Mini is shown, but a Nano will also work. Pro-mini's pinout may vary. Programming hearder sometimes reversed.

3) Most external items are optional, besides speed pot and paddle

4) Auto-character space (ACS) is OFF when switch is open

5) Can be powered from 7 to 12 V to RAW pin

6) Ground A1 if no pot is used, to give default speed

Accu-Arduino Keyer		
WA5BDU	Rev 9 FEB 2024	Page 1 of 1