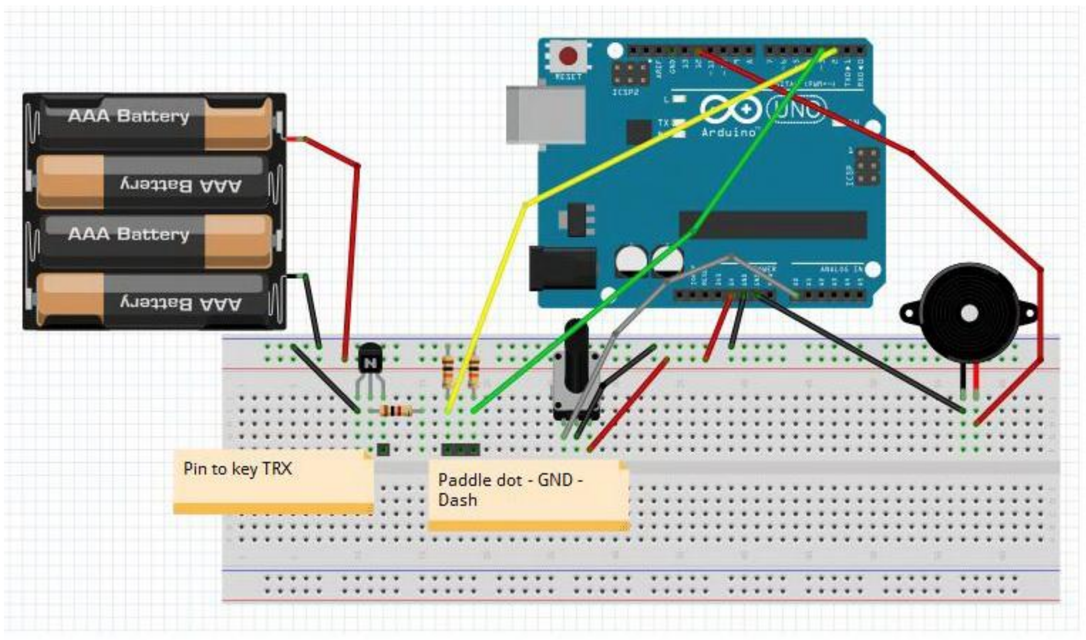
Simple Arduino Morse Keyer

By Kevin Mc Donald ZS6KMD

Welcome to our first Arduino project in the Radio ZS magazine. I hope to entertain you with some handy tips, ideas on what to build and even maybe help you add some interesting devices to your shack. This month we will start by looking at how to build a simple Arduino based Morse Keyer.

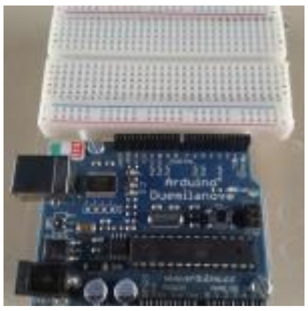
What you will need to build this project is an Arduino (almost any model will work with this project), a 10K potentiometer, a 1K resistor, two 10K resistors, a BC237 transistor, a piezo buzzer, a breadboard and some jumper wires for the breadboard.

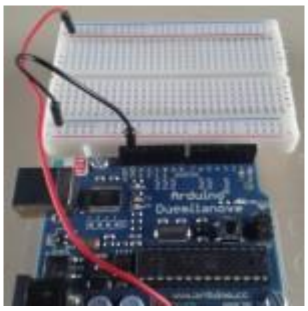
Below is the Fritzing layout of the Morse Keyer.

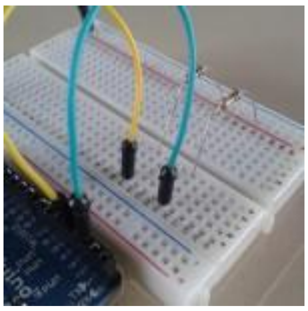


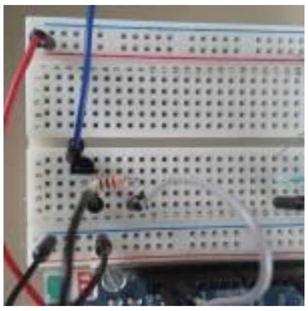
How To:

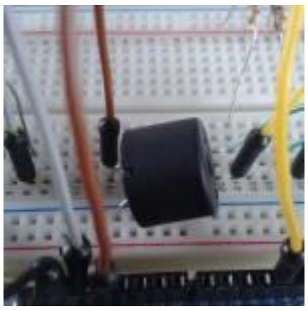
We start with our Arduino, and create some inputs and outputs, in this case, we will be using 2 Digital inputs for the Dot and Dash, 1 Digital output to key the transceiver, 1 Digital output to sound the buzzer and 1 Analogue input to read the speed from the potentiometer.

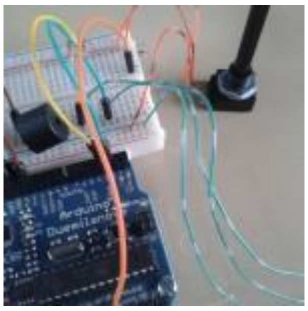

We start with our Arduino & Breadboard


Connect the power rails of the Breadboard to the +5V and GND on the Arduino


Insert the two 10K resistors and connect to D2 and D3 on the Arduino


Insert the BC237 (note pinouts) with the 1K resistor to the base, emitter to GND and collector to TRX


Connect the Piezo buzzer between D12 and GND


Connect the 10K potentiometer with pin1 to +5V, pin2 to A0 and pin3 to GND

Now we can move on to the software part of our Morse Keyer.

Writing "code" for Arduino need not be daunting; there are simple steps to remember. So let us start with the first of 4 steps, making Declarations.

We define the constants for the pins that we will use and also the variable (integer) for the value of the potentiometer.

```
#define P_DOT    2   // Connects to the dot lever of the paddle

#define P_DASH   3   // Connects to the dash lever of the paddle

#define P_AUDIO 12   // Audio output

#define P_CW    13   // Output of the keyer, connected to your radio

#define P_SPEED A0   // Attached to centre pin of potentiometer which allows you to set the keying speed
```

Nest we do the setup routine, this is run only once at start-up and here we define which pins are used as inputs and outputs using the *pinMode* statement. We also ensure that the key is not always on by setting the output pin D13 to LOW using the *digitalWrite* statement.

```
// Initializing the Arduino

void setup()

{

  pinMode(P_DOT, INPUT);

  pinMode(P_DASH, INPUT);

  pinMode(P_AUDIO, OUTPUT);

  pinMode(P_CW, OUTPUT);

  digitalWrite(P_CW, LOW);     // Start with key up

}
```

You will notice that we have used the constants that we declared above and not the pin numbers.

Next we move on to the main routine called *loop*, which repeats itself continuously until the program is terminated or an alternative instruction is received.

Our first statement reads the value of our Analogue input (potentiometer) to tell us what speed the keyer is set at and stores it as our *speed* variable. Next we read the input pin connected to the dot lever of the Morse paddle using *digitalRead*. If the pin is "low" we call a function called *keyAndBeep*, then there is a short delay as we require a time between dots and dashes of one dot length and finally we do the same for the dash taking note that we need the key to be down three times longer using the *speed*3* argument. The routine then returns to the beginning and reads the first statement again for the potentiometer value and so on.

```
// Main routine

void loop()

{

 speed = analogRead(P_SPEED)/2; // Read the keying speed from potmeter

 if(!digitalRead(P_DOT))      // If the dot lever is presssed..

 {

  keyAndBeep(speed);         // ... send a dot at the given speed

  delay(speed);              //   and wait before sending next

 }

 if(!digitalRead(P_DASH))      // If the dash lever is pressed...

 {

  keyAndBeep(speed*3);        // ... send a dash at the given speed

  delay(speed);              //   and wait before sending next

 }

}
```

The last section deals with keying the transceiver and making the beep sound. This is called *keyAndBeep*. It starts by setting pin D13 to high which will drive the transistor to conduct and key the transmitter. Next we have a *for* loop which runs for a while, all statements contained in this loop are repeated continuously depending on the speed value of the potentiometer. At each iteration we set the value to high, wait 1 millisecond and set it to low and so on. The resultant output of this is roughly a 500hZ square wave. When this loop ends, the pin D13 is set to low once more and the transmitter will stop keying.

```
// Key the transmitter and sound a beep

void keyAndBeep(int speed)

{

 digitalWrite(P_CW, HIGH);          // Key down

 for (int i=0; i < (speed/2); i++)    // Beep loop

 {

  digitalWrite(P_AUDIO, HIGH);
```

```
    delay(1);

    digitalWrite(P_AUDIO, LOW);

    delay(1);

  }

  digitalWrite(P_CW, LOW);          // Key up

}
```

We have now completed the code for our Morse Keyer. If you have missed putting it together, you can download the entire code at http://www.zs6kmd.za.net/Arduino_Morse_Keyer.txt

You can now connect the Arduino to your PC, open the Arduino software, select your board type and com port and program the Arduino. Once programmed you can disconnect from the PC and connect the power. You should now be able to hear the dots and dashes if you press the paddle.

Congratulations on building your first Arduino project with us, you can build the circuit on a small piece of Vero board and put it into a case or just break it all down, ready for the next project. Remember, you can add additional features such as a flashing LED, make it smaller using the Arduino Nano or Pro Mini etc.

Happy Building

73 de Kevin ZS6KMD