WD9GYM Morse Code Decoder for the Ten Tec Patriot

By Marty Boroff WD9GYM

Background

I was first licensed in 1977. I operated a Kenwood TS 820 until 1985. After earning my Advance class license I attempted to break the 15 wpm barrier so that I could earn an Extra class license. In 1985 I became attached to a project that deployed the IBM PC. I drifted away from Amateur radio and became immersed in PC programming.

I retired in 2006 and did not do any computer programming until 2014 when I stumbled into the Arduino world. In 2015 I read the book “Arduino Projects for Amateur Radio” by Jack Purdum and Dennis Kidder. Their projects on a Morse Code decoder and PS/2 Keyboard keyer along with the impending release of the Ten Tec Patriot convinced me to re-enter the hobby.

I joined the Patriot group on Yahoo. Upon receipt of my Patriot I began making coding changes to Ten Tec’s release code along with other members of the group. We formed the Patriot Alliance to provide a release of software with enhancements desired by different members of the group. This project is another offshoot of the code and hardware.

First attempts

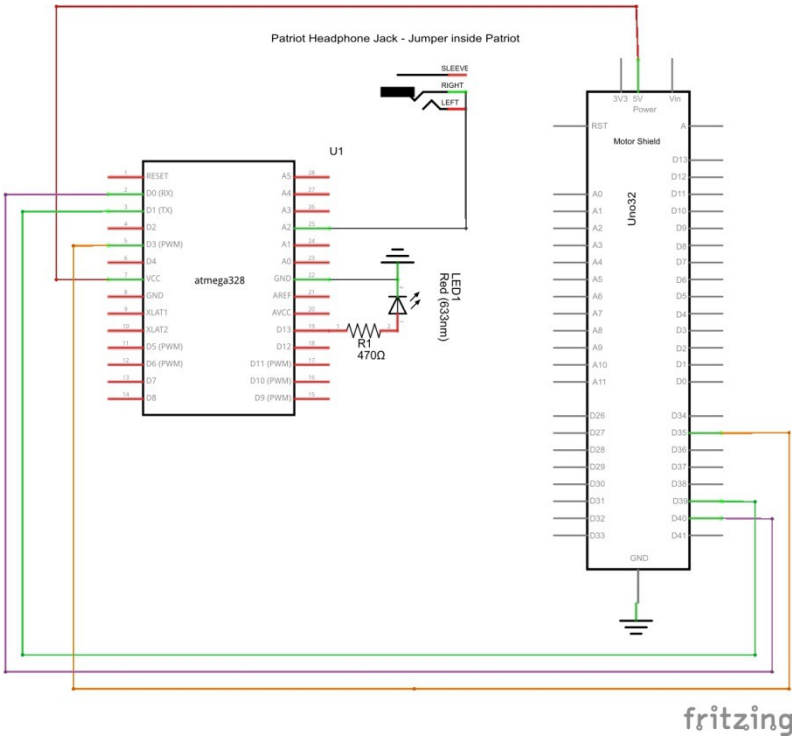
At first I attempted to employ the decoder I had built from “Arduino Projects for Amateur Radio” I found that it was difficult to tune using the designed 50 K ohm trimmer. Placing it in an enclosure also seemed difficult. Next I purchased an inexpensive decoder on eBay. It worked but had tuning failures also as it uses a 10K ohm trimmer to tune. Both of these units also would not have been able to integrate into the Patriot enclosure.

I also attempted to integrate different version of decoder software in the Patriot Alliance code. It became evident that decoder software requires 100% cpu time in order to accurately decode the tones. In search for decoder software I stumbled upon EASY BUILD CW DECODER BASED UPON DSP GOERTZEL CODE http://www.skovholm.com/cwdecoder. I quickly was able to determine that this code worked well on a Uno R3 directly connected to the audio jack used for headphones or speaker.

Final Solution

The final solution was to develop a bare bones ATmeg328P on a Uno32 prototype board that was already in use for a 20x4 LCD with the Patriot. This reminded me of the early PC days when co-processors were used to enhance number crunching. In short I was able to process the decoding on the Arduino and send the decoded character to the Uno32 processor in the Patriot over serial interfaces. Decoding is activated by holding the Selection button for greater than .5 seconds. Frequency range of decoding is 570 – 810 Hz. Using a PC to generate tones testing a max speed of 80 wpm was achieved. Simulating the addition of noise and lowering signal strength the low end of decoding was achieved at S5 on noise level and S3 on signal level.

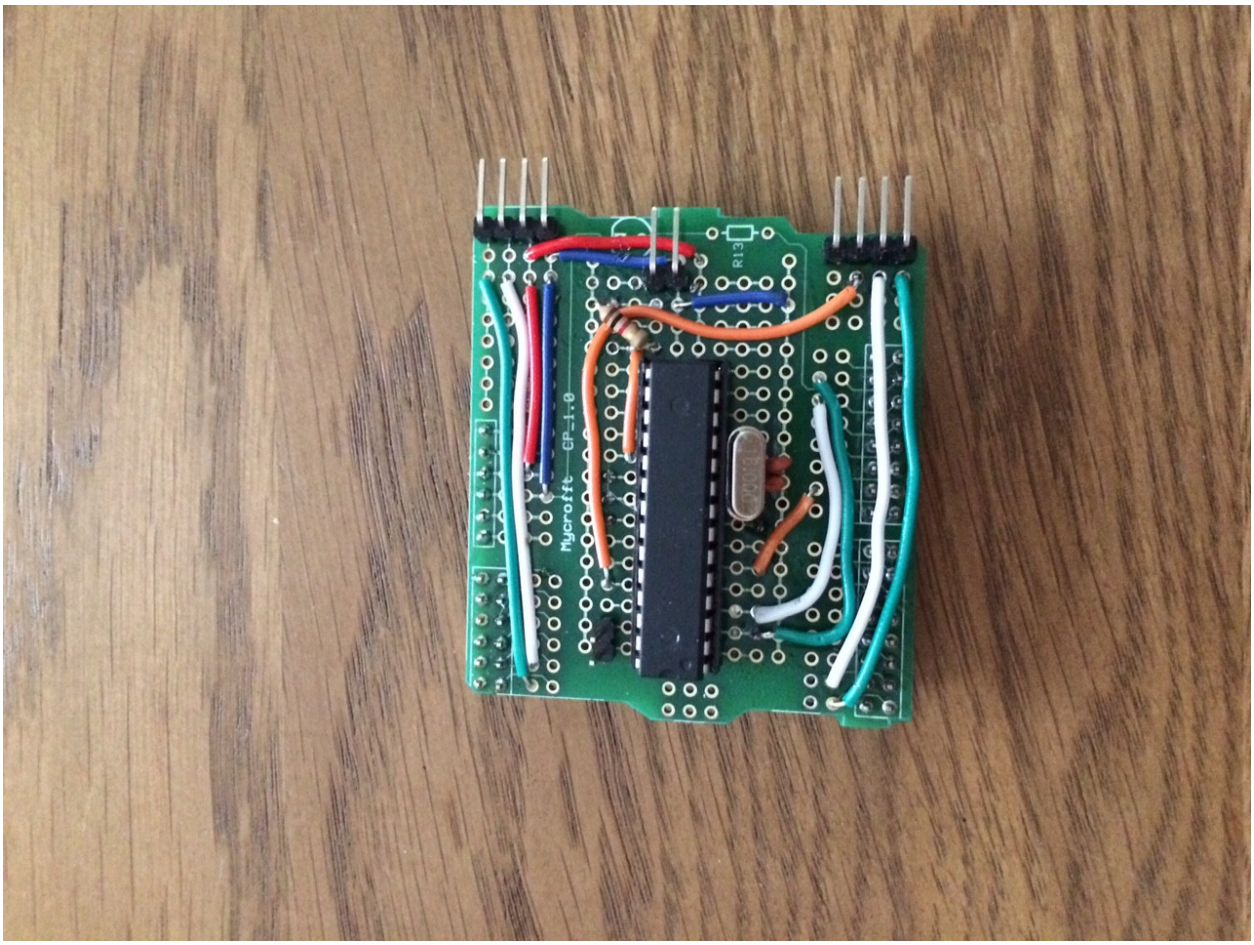
Curcuit



Circuit for connecting the ATmeg328P and Uno32

Hardware

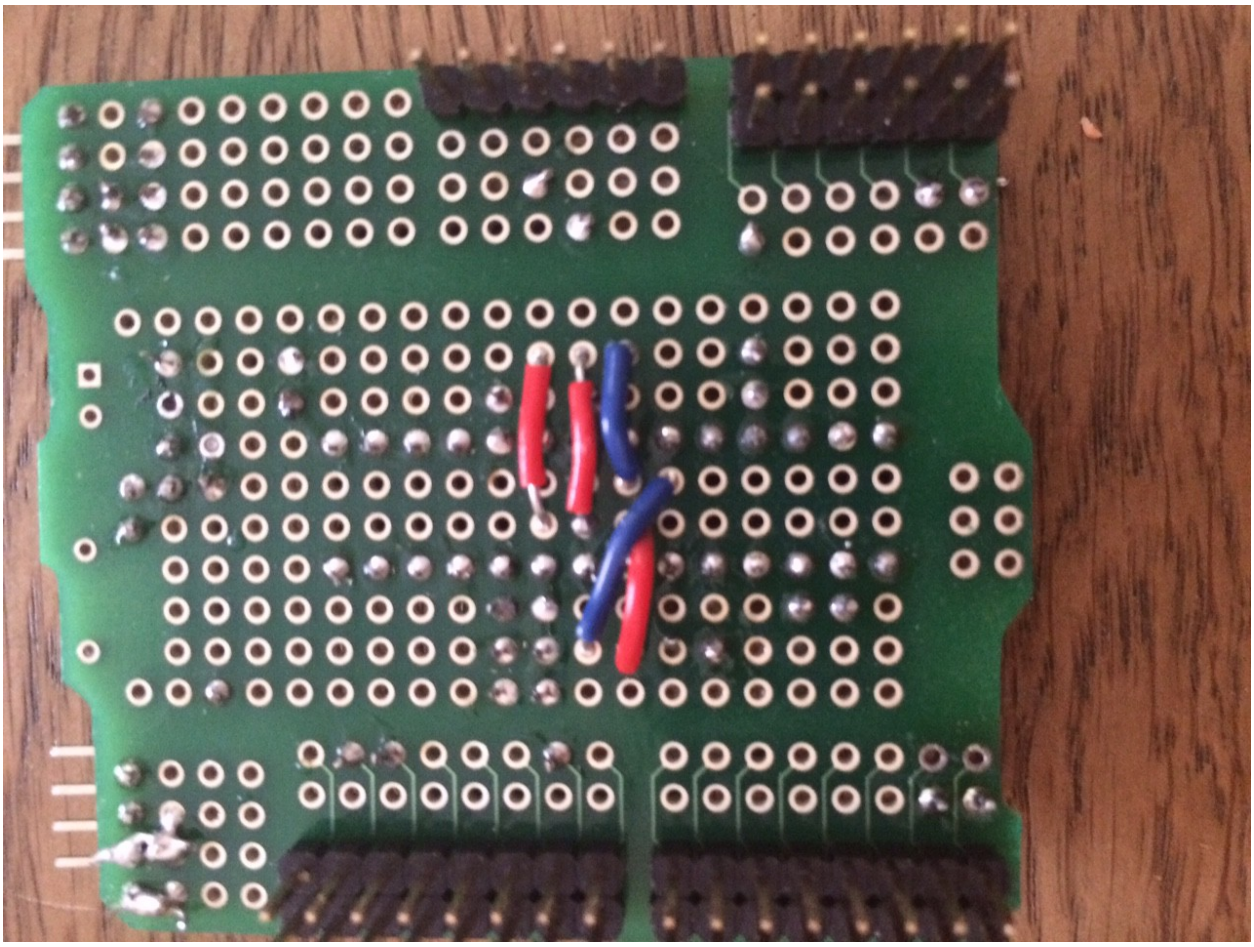
A bare bones ATmeg328P requires two 22uF capacitors and on 16Mhz crystal. Vcc and Gnd are provided by the Uno32.An ATmeg328p draws 0.2 mA when active.



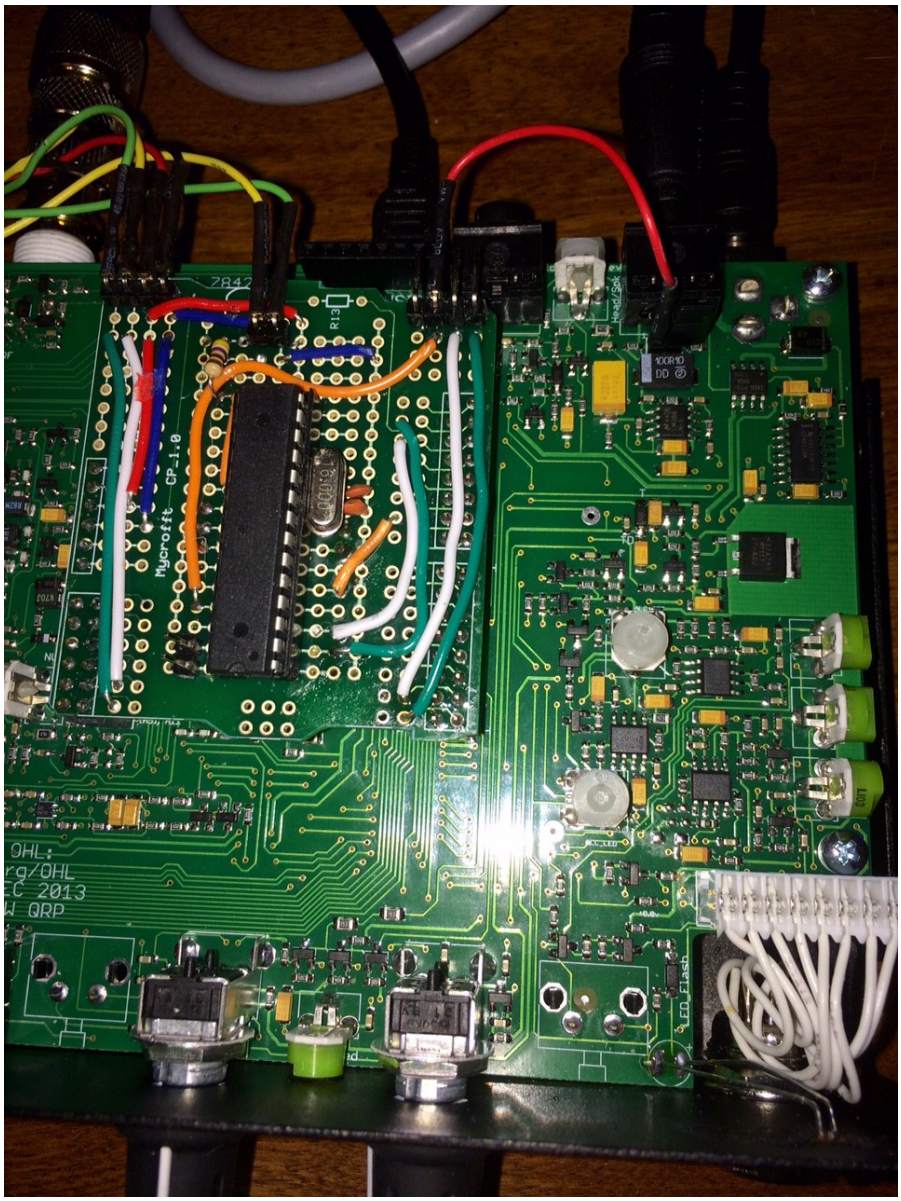
ATmeg328P mounted on a Uno32 Prototype Board

The pins on the left are connected to a 20x4 LCD. The pins in the center connect to the tuning LED. The orange wire connect digital pin 3 on the ATmeg328P to digital pin 35 on the Uno32, used to signal when to decode. The short green and white pair connect the serial pins on the ATmeg328P to the Serial 1 pins on the Uno32. The longer pair of green and white wires connect the serial pins of the Uno32 for a future project. The long orange wire connects pin A2 on the ATmeg328P to the third jumper from the right on the pins at the top right to connect to the audio jumper on the Patriot.

A diagram of theATmeg328P pins is located at the end of this document. The crystal is connected to pins 9 and 10 of the ATmeg328P. A 22uF capacitor is connected between pin 9 and gnd and pint 10 and gnd. The LED anode is connected to a 470 Ohm resistor to pin 13 and the cathode is connected to gnd.



The red wires carry Vcc and the blue wires carry Gnd.



Prototype board mounted in the Patriot

The red wire in the upper right corner connects to the left jumper pin for speakers and headphones.

Software

All software is located in the Patriot Alliance Mod folder in the Yahoo Ten Tec Patriot group files under the name of XXXXXX.ZIP.

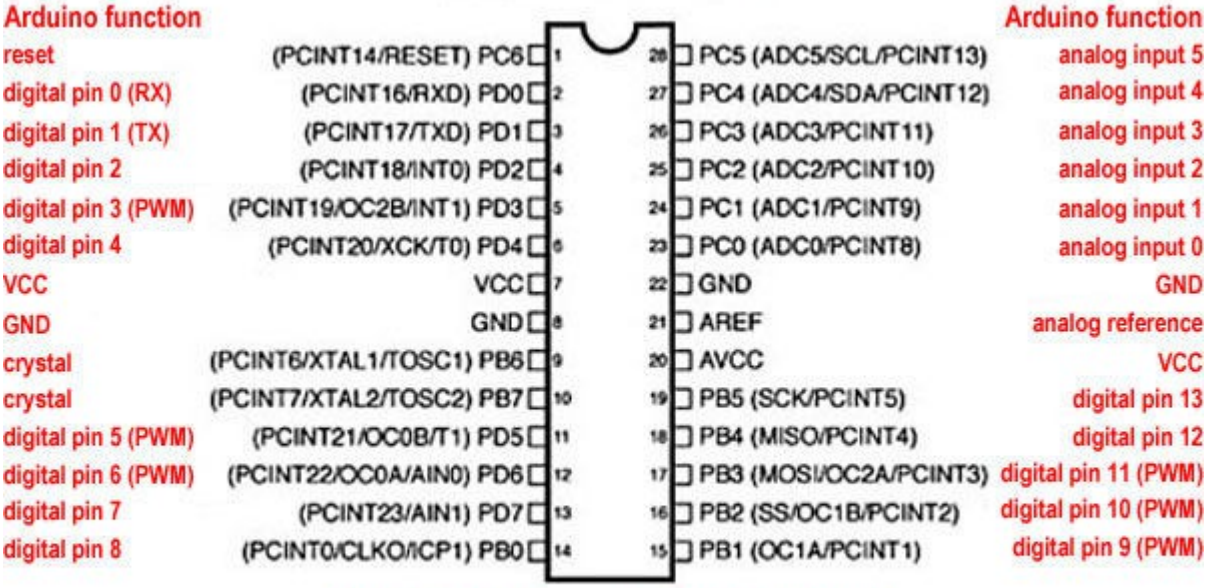
A list of the steps I went through deriving the solution are documented in the beginning of the Patriot code. These include attempts to integrate the decoding software. Most important are the steps where current operating information is moved to the first three lines of the display to make room for the decoding characters to scroll across the fourth line.

Code snippets for the Patriot code are:

```
void loop()
{
 TX_routine();
              .
              .
              .

/*************************************************
    decoder decoder loop
*************************************************/
   if (enableDecoder == true) {
      if (Serial1.available() > 0){
         ch = Serial1.read();
         for (int i = 0; i < 19; i++) {
             lcdBuff[i] = lcdBuff[i + 1];
         }
      lcdBuff[19] = ch;
      lcd.setCursor(0,3);
      lcd.print(lcdBuff);
      }
   }
     .
     .
  }
}   //  END LOOP

================================================================================

void  Selection()
{
   Step_Select_Button = digitalRead(Select_Button);
   if (Step_Select_Button == HIGH)
   {
     // Debounce start
     unsigned long time;
     unsigned long start_time;
     unsigned long long_time;
     long_time = millis();
    time = millis();
     while( digitalRead(Select_Button) == HIGH ){
       // function button is pressed longer then 0.5 seconds
       if ( (millis() - long_time) > 500 ) {
#ifdef DECODER
/*********************************************************
         decoder flip
*********************************************************/
         // flip logic boolean enableDecoder
          if (enableDecoder == true) {
             enableDecoder = false;
             digitalWrite(switchPin, LOW);
             lcd.setCursor(0, 3);
             for (int i = 0; i < 20; i++) {
                 lcd.print(" ");
             }
             lcd.setCursor(14, 2);         // print the decode flash on line 3
```

```
            lcd.print("     ");        // blank out dot
            lcd.setCursor(7, 3);
            lcd.print("PAM 1.2");
          } else {
            enableDecoder = true;
            digitalWrite(switchPin, HIGH);
            lcd.setCursor(14, 1);         // print the decode flash on line 3
            lcd.print("     ");        // blank out S meter bars
            lcd.setCursor(0, 3);          // clear last line
            for (int i = 0; i < 20; i++) {
                lcdBuff[i] = ' ';
                lcd.print(" ");
            }
          }
/*******************************************************
        end of decoder flip
*******************************************************/
#else
         // announce frequency
         TX_frequency = (frequency_tune - IF)/100;
        char buffer[8];
         ltoa(TX_frequency, buffer, 10);
         announce(buffer);
#endif
         // wait for button release
         while( digitalRead(Select_Button) == HIGH ){
         }
         return;
        }
        start_time = time;
        while( (time - start_time) < 7) {
          time = millis();
        }
      } // Debounce end
      Step_Select_Button1 = Step_Select_Button1++;
      if (Step_Select_Button1 > 2 || (Step_Select_Button1 > 1 && Step_Multi_Function_Button1 == 2) )
      {
         Step_Select_Button1 = 0;
      }
   }
   Step_Select();
}
```

ATmega328 Pin Mapping



Degital Pins 11, 12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega 168 pins 17, 18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.